UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Desarrollo de una aplicación esteganográfica para Android

Autor: Sergio Pérez Olivares
Tutor: José Miguel Lozano Barceló

Colmenarejo, 8 de Febrero de 2013

Agradecimientos

En primer lugar, agradezco a mis padres toda la paciencia que han tenido conmigo, especialmente durante estos meses difíciles, en los que me he visto superado en algunas ocasiones por las circunstancias. Siempre he recibido su apoyo y comprensión, nunca sabré como agradecérselo. A mi hermano Javi, al que no he podido dedicar todo el tiempo que se merece y que yo hubiese deseado.

A mis amigos de clase, con los que he pasado cuatro años de momentos increíbles, llenos de risas, de situaciones al límite y agobios antes de cada entrega. Especialmente a Carlos, Nico, Quique e Iñaki, sin ellos estos años de universidad no hubieran sido lo mismo. Espero que los lazos de amistad que hemos forjado durante esta etapa de mi vida no se rompan jamás.

A mi tutor José Miguel, cuyos consejos y ayuda me han servido de motivación para realizar este proyecto.

A mi abuela Rogelia y a mi primo Eduardo, a los que siempre llevaré en mi corazón.

Resumen

En este proyecto se propone el desarrollo de una aplicación para dispositivos móviles Android, que permita mantener comunicaciones seguras y ocultas a través de los programas de mensajería instantánea más populares de la actualidad.

Muchas aplicaciones de mensajería instantánea no poseen las medidas de seguridad y privacidad necesarias que permitan a los usuarios intercambiar información sensible sin que ello entrañe un riesgo importante.

El sistema propuesto es capaz de introducir la información secreta que se quiera comunicar dentro de un medio digital tan inocente como es una imagen. De esta forma, los usuarios sólo tienen que transferirse una imagen para intercambiar mensajes secretos, sin levantar ninguna sospecha y de manera completamente discreta a ojos de un atacante.

La información que se oculta en las imágenes estará protegida con los algoritmos criptográficos más seguros, de modo que el emisor y receptor sean los únicos que puedan recuperar el mensaje original empleando una contraseña común, la clave de cifrado.

Este sistema supone un avance muy importante para la esteganografía en dispositivos móviles, ya que es el primero que permite enviar imágenes esteganografiadas a través de mensajería instantánea sin que el mensaje secreto quede dañado por los mecanismos de compresión que se emplean.

La compresión de imágenes ha sido siempre el punto más débil de los programas esteganográficos existentes. Sin embargo, el aquí propuesto emplea mecanismos de ocultación y recuperación de errores que permiten que el mensaje no se pierda tras la compresión que se realiza antes del envío de la imagen.

El programa desarrollado será compatible con las aplicaciones más utilizadas de mensajería instantánea, éstas son: WhatsApp, Line y Spotbros. Asimismo la aplicación también funcionará con los sistemas de mensajería tradicionales como son el correo electrónico y el MMS.

Palabras clave: Esteganografía, imágenes, mensajería instantánea, Android, WhatsApp, Line, Spotbros, email, MMS, seguridad, criptografía.

Abstract

This project proposes the development of an application for Android mobile devices, it will allow to keep hidden and secure communications through the most popular instant messaging programs.

Many instant messaging applications don't have the needed security and privacy measures to allow users to exchange sensitive information without compromising their data.

The proposed system is able to introduce the desired secret information into such a common digital medium as are images. In this way, users only have to send the image in order to exchange secret messages, it won't be noticeable for any eavesdropper.

The information that is hidden into the pictures will be protected with the most secure cryptographic algorithms, so the sender and the receiver will be the only ones able to recover the original message using a shared password, the encryption key.

This system represents an important improvement for steganography on mobile devices as it's the first one that allows sending steganographic pictures through instant messaging without getting damaged the secret message by the employed compression mechanisms.

Image compression has always been the weakest point of the existing steganographic programs. However, the one proposed here uses error recovery mechanisms allowing the message not getting damaged after compression in the image sending process.

The developed program will be compatible with the most employed instant messaging applications, these are: WhatsApp, Line and Spotbros. In addition, it will also work with the traditional messaging systems such as email and MMS.

Keywords: Steganography, pictures, images, instant messaging, Android, WhatsApp, Line, Spotbros, email, MMS, security, cryptography.

Índice

Índice de Figuras

Índice de Tablas

1 INTRODUCCIÓN

1.1 CONTEXTO ACTUAL

Las aplicaciones de mensajería móvil se han convertido en uno de los medios de comunicación más usados en la actualidad. Estas aplicaciones permiten el envío de mensajes de texto y archivos multimedia entre distintos usuarios simultáneamente a través de Internet.

Sin embargo, estos sistemas de comunicación carecen en muchos casos de las medidas de seguridad y privacidad necesarias, lo que las sitúa como un objetivo muy apetitoso para posibles atacantes. Para los usuarios es un riesgo enviar y recibir información sensible a través de estas aplicaciones, ya que sus comunicaciones pueden ser interceptadas por terceros.

La criptografía ofrece la posibilidad de cifrar la información entre el emisor y el receptor, de manera que sin la clave de cifrado empleada nadie sea capaz de observar la conversación.

Han sido muchas las aplicaciones criptográficas que se han sugerido como complemento para proteger la información que se transfiere en este tipo de medios. Sin embargo, está aproximación supone dos inconvenientes principales:

- El empleo de criptografía está prohibido en muchos países del mundo, como Francia o China. En otros como Estados Unidos está fuertemente controlado al ser considerados como una amenaza para la seguridad nacional [1].
- El uso de mensajes cifrados denota la importancia de la información intercambiada, lo que llamaría tremendamente la atención de un atacante, que podría intentar obtener la clave de descifrado o alterar el mensaje transmitido para que el receptor no fuera capaz de recuperarlo.

En este contexto, se hace necesario el empleo de una nueva técnica que permita intercambiar mensajes sin llamar la atención de terceros y con toda la seguridad de la criptografía, esta técnica es conocida como esteganografía. La esteganografía es capaz de ocultar información secreta en cualquier tipo de medio digital, como son las imágenes, para que no pueda ser detectada ni interceptada fácilmente.

Actualmente existen diferentes aplicaciones para dispositivos móviles Android que permiten el empleo de esteganografía en imágenes. Sin embargo, todas ellas tienen un problema en común, y es que cuando se envían las imágenes a través de canales que imponen compresión es imposible recuperar posteriormente el mensaje oculto. Este es el caso de las aplicaciones de mensajería instantánea.

1.2 MOTIVACIÓN

La motivación del proyecto surge tras comprobar que las aplicaciones esteganográficas para Android no funcionan correctamente cuando se transfieren las imágenes a través de mensajería instantánea. Esto es porque las imágenes antes de ser enviadas sufren una compresión tan grande que el mensaje secreto queda dañado, de forma que tras su recepción es imposible extraerlo.

Creo que es una idea muy original para un proyecto porque se va a intentar conseguir lo que otras aplicaciones no han logrado antes, y es poder tener comunicaciones secretas y ocultas a través de programas como WhatsApp, Line o Spotbros, que son la referencia actual en el ámbito de la mensajería instantánea. Asimismo se intentará que los sistemas tradicionales de mensajería como el correo electrónico y el MMS sean compatibles con la aplicación que se va a desarrollar.

Durante el presente proyecto va a ser necesario:

- Realizar un estudio en profundidad de los métodos de esteganografía en imágenes existentes.
- Comprender el proceso de compresión de imagen y cómo se pueden minimizar sus efectos.
- Analizar el funcionamiento de los programas de mensajería instantánea.
- Desarrollar una aplicación compatible con la plataforma Android que sea capaz de interactuar con el usuario para ocultar y extraer mensajes en imágenes.

Finalmente tras el desarrollo del proyecto, se comprobará la utilidad real de la aplicación poniéndola a disposición de los usuarios en *Google Play*. Dependiendo de los resultados que se obtengan se propondrán distintas líneas de futuro para mejorar la aplicación.

1.3 OBJETIVOS

El objetivo principal del presente proyecto es construir una aplicación para dispositivos móviles Android que, haciendo uso de la esteganografía en imágenes, permita al usuario mantener comunicaciones **seguras** y **discretas** en aplicaciones de mensajería instantánea y correo electrónico.

Para alcanzar dicho fin, la herramienta debería ser capaz de cumplir los siguientes sub-objetivos:

- La aplicación tiene que tener dos modos de operación: uno para ocultar mensajes y otro para extraerlos.
- Proteger el contenido de los mensajes utilizando algoritmos criptográficos seguros. De modo que éstos no puedan ser descifrados aunque un atacante detecte su presencia.
- Dispersar la información de manera pseudo-aleatoria por la imagen para dificultar su detección y percepción visual.
- Las imágenes resultantes deben ser lo suficientemente robustas como para poder soportar compresiones y corregir errores introducidos durante su transmisión.
- El programa tiene que ser compatible con alguna aplicación de mensajería instantánea como WhatsApp, y también con las aplicaciones de correo electrónico convencionales.

Adicionalmente se consideran como deseables los siguientes objetivos:

- Introducir mensajes de hasta 1000 caracteres de longitud en el mensaje.
- Permitir la ocultación de información en imágenes de cualquier formato y resolución.
- Compatibilidad con la mayoría de dispositivos y versiones del sistema operativo Android.
- Compatibilidad con otras aplicaciones de mensajería instantánea como Line y Spotbros.
- Las imágenes resultantes deberían ser indetectables usando software de estegoanálisis.
- La repercusión visual introducida en la imagen portadora del mensaje no debería ser muy elevada.
- Hacer pública la aplicación en *Google Play* para analizar y comparar los resultados obtenidos con aplicaciones que ofrecen prestaciones similares.

2 ESTADO DEL ARTE

En este capítulo se introducen las principales bases sobre las que se sustenta el presente Trabajo de Fin de Grado. Se considera esencial la comprensión de los siguientes conceptos para el posterior desarrollo de la aplicación propuesta:

2.1 ESTEGANOGRAFÍA

2.1.1 DEFINICIÓN

El término 'esteganografía' procede de dos palabras griegas, *steganos*: oculto, encubierto; y *graphos*: escritura. La esteganografía es definida por Markus Kahn como "*el arte y la ciencia de comunicarse ocultando la existencia de la propia comunicación*" [2]. En otras palabras, la esteganografía es la ciencia de transmitir información oculta, de forma que sólo el emisor y receptor son conscientes de que se está llevando a cabo un envío de información [3].

La esteganografía utiliza un medio digital (archivos de texto, audio, imagen, vídeo, etc.) como archivo de transporte, para ocultar la información, a este medio se le conoce como contenedor o portador y su objetivo es no levantar sospechas; del mismo modo el mensaje secreto o la información a ocultar puede ser cualquier medio digital. Una vez se oculta el mensaje en el archivo contenedor utilizando un técnica esteganográfica se obtiene un *esteganograma*, que contendrá el mensaje oculto en el archivo portador. En la figura 1 se muestra el proceso esteganográfico básico: el mensaje se oculta en el contenedor para formar el esteganograma, que se puede transferir por un medio de comunicación inseguro para posteriormente recuperar el mensaje aplicando el proceso inverso [4].

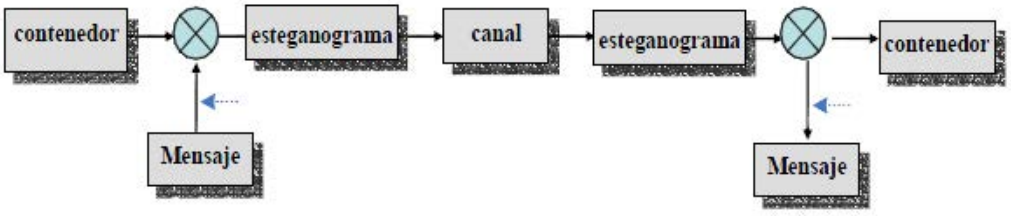


Ilustración 1 Proceso esteganográfico básico [4]

2.1.2 RESEÑA HISTÓRICA DE LA ESTEGANOGRAFÍA

La esteganografía no es una técnica que se haya inventado recientemente, de hecho data de los tiempos de la antigua Grecia, donde disponer de cierta información podía significar la diferencia entre la vida y la muerte.

Se cuenta habitualmente la historia de cómo *Demeratus* utilizó la esteganografía para comunicar a la ciudad de Esparta que *Jerjes* tenía planes de invadir Grecia. Para evitar ser capturado por espionaje en los controles, escribió sus mensajes en tablas que luego fueron cubiertas con cera (ver figura 2), de manera que parecían no haber sido utilizadas.

Ilustración 2: tablas cubiertas de cera [5]

Asimismo, durante la misma época se cuenta otro método esteganográfico para pasar información, consistía en rasurar la cabeza de un individuo y tatuar en ella un mensaje, posteriormente cuando el pelo le volviera a crecer el mensaje tatuado permanecería oculto.

Sin embargo no fue hasta principios del siglo XVI cuando un alemán, *Johannes Trithemius*, publicará el primer libro que tratará sobre la esteganografía ("*Steganographia*", ver figura 3). Dicho libro hablaba de la ocultación de mensajes entre otros muchos temas. A partir de dicha publicación comienzan a aparecer más libros con temática referente a la ocultación de la información.

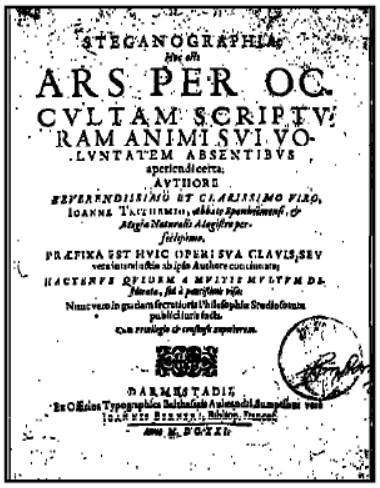


Ilustración 3: Primer libro sobre la esteganografía [5].

A partir de la Segunda Guerra Mundial se desarrollaron nuevos métodos para esconder las comunicaciones. Uno de estos métodos consistía en ocultar código Morse en los signos de

puntuación de ciertas letras, de modo que se podía mantener una conversación confidencial sin ser descubierto. Otra de las técnicas más ingeniosas y usadas fue el "*cifrado nulo"*, que consistía en enviar un mensaje cualquiera y elegir cierta parte del mismo para ocultar un mensaje privado.

Tampoco podemos olvidarnos de métodos tan tradicionales como las tintas invisibles, que utilizaban ciertas sustancias para ocultar el mensaje y posteriormente ser descubiertas con calor.

Hoy en día todas estas técnicas han pasado a un segundo plano. Con la explosión de la era digital es posible ocultar mensajes secretos en objetos tan comunes como las imágenes, que es lo que vamos a explicar en siguientes apartados.

2.1.3 USOS DE LA ESTEGANOGRAFÍA

Existen muchas formas en las que podemos emplear la esteganografía hoy en día, una de las más comunes es a través de las imágenes. También hay muchas razones por las que necesitamos emplear este arte de ocultación de mensajes, algunos de estos propósitos pueden ser buenos y otros no tanto. Así pues, y a modo de ejemplo, paso a enumerar posibles usos de esta técnica:

- ***Watermarking* o marcas de agua digitales**: esta técnica sirve para guardar información sobre el autor o propietario intelectual del fichero digital.
- ***Fingerprinting* o huella digital**: Se oculta información sobre el propietario que adquirió los derechos y la fecha en la que lo hizo. De esta forma es posible controlar el uso inadecuado del fichero.
- **Autenticación**: Se puede emplear como archivo de autenticación, donde la clave es el mensaje ocultado en su interior.
- **Estructuras de programación**: Puede emplearse para albergar estructuras especiales de datos, esto puede llegar a ser realmente útil para programadores y desarrolladores.
- **Metadatos**: En fotografías puede resultar realmente útil para albergar información descriptiva referente a la imagen.
- **Información personal:** Se puede proteger la información personal de un individuo para evitar ser explotada por terroristas, delincuentes *hackers*, etc.
- **Comunicaciones ocultas**: A priori el uso más lógico. Se puede usar para ocultar importantes comunicaciones privadas. La transmisión de ficheros encriptados puede llamar la atención de terceros, esto puede evitarse utilizando esta técnica.

A pesar de todo ello, los usos más empleados de la esteganografía en imágenes acaban siendo los menos deseados: terrorismo, pornografía infantil, narcotráfico y estafas. Estos individuos emplean la esteganografía como una herramienta diaria más con fines malignos.

Así por ejemplo, para referenciar un caso particular que muestra la importancia de este arte; se dice que las órdenes y coordinaciones de los atentados del 11 de Septiembre de 2001 se ultimaron a través de la esteganografía, incrustando la información en imágenes pornográficas, las cuales nunca serían rastreadas por los organismos de inteligencia de Estados

Unidos, debido a que la religión de estos terroristas les prohíbe ver a mujeres desnudas. También se dice que las comunicaciones se llevaban a cabo desde el borrador de una cuenta de correo, esto es porque los mensajes guardados como borrador no son enviados por Internet, por lo que no se pueden rastrear [6].

Hoy en día otro de los ejemplos más recientes y sofisticados del uso de la esteganografía en imágenes es la aplicación en troyanos. Se tiene consciencia de que existen actualmente varios tipos de troyanos bancarios que, una vez instalados en el sistema de la víctima, son capaces de intercambiar la información con el atacante a través de imágenes públicas subidas en diversos blogs de Internet. De esta forma el atacante sólo tiene que ir recolectando periódicamente estas imágenes distribuidas y extraer la información que ha sido ocultada esteganográficamente en ellas. Es un sistema bastante interesante pues el atacante no mantiene ningún tipo de contacto directo con el sistema infectado que le pueda incriminar como culpable.

Antes de finalizar la presente sección, me gustaría comentar que la esteganografía no es la culpable de los hechos delictivos citados anteriormente, no hay medios culpables, sino personas culpables [7]. Y tenemos que aprender a usar razonablemente las ventajas que nos da esta técnica y combatir todos aquellos usos que traspasen la férrea línea de la legalidad.

2.1.4 TÉCNICAS ESTEGANOGRÁFICAS

La esteganografía puede ser aplicada en todo tipo de medios digitales como son: textos, audio, imágenes, vídeo, sistemas de ficheros, etc. En la presente sección haremos un repaso general de las técnicas más empleadas en imágenes, ya que el proyecto consiste en ocultar información en este tipo de medios.

Las técnicas esteganográficas básicas sobre imágenes se clasifican dependiendo del mecanismo o principio empleado para ocultar el mensaje secreto. Se distinguen 7 técnicas elementales:

SISTEMAS DE SUSTITUCIÓN

Se trata de sustituir ciertos bits del archivo contenedor (en nuestro caso una imagen) por los de la información a esconder. La principal ventaja que encontramos es que el tamaño del fichero no difiere en exceso respecto al original. Además, gracias a la redundancia y/o exceso de detalle en dichos ficheros, la calidad permanece intacta.

Cuando trabajamos con imágenes, el método tradicional de sustitución de bits se centra en aquellos que son menos significativos. Es decir, si sustituimos sólo los bits con menor importancia de cada píxel, obtendremos pocas variaciones en el resultado final. Por el contrario si empleamos los bits más significativos para insertar información, la imagen portadora quedará bastante más distorsionada, y esto será detectado fácilmente por los humanos y computadores.

Al método anterior se le denomina popularmente **LSB** (*least significant bit*). A modo de ejemplo veremos qué importancia tiene la modificación del bit con menos valor de un píxel en

una imagen. Dada una imagen que emplea 8 bits para representar cada píxel, los valores posibles oscilan entre 0 y 255. El bit más significativo contribuye en 128 unidades al número total si es un 1. Esto quiere decir que el bit menos significativo puede cambiar la intensidad de un píxel en la imagen final entre el 0.5% y 1% como mucho. Incluso empleando los dos últimos bits de cada píxel, cuya aportación final es de 3 unidades como mucho, podemos conseguir resultados imperceptibles para el ojo humano. En la figura 4 se muestra la imperceptible diferencia variando en un bit cada una de las componentes RGB de un píxel.

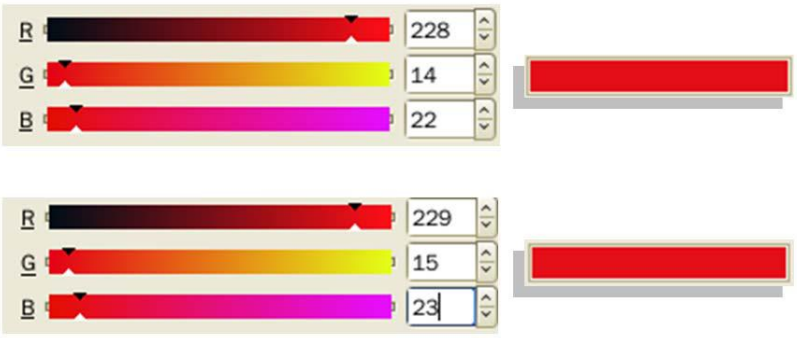


Ilustración 4: Percepción visual de la modificación de los LSB de un píxel [8].

Esta técnica funciona mejor cuando el archivo de imagen es grande y posee fuertes variaciones de color, aparte de mejorar también su efectividad cuanto mayor sea la profundidad del color [3]. Resulta especialmente sencillo aplicar este método sobre imágenes BMP (mapa de bits), que no están comprimidas, ocupan un gran tamaño y tienen una paleta de colores notable.

La principal desventaja de esta técnica es que asume que la información almacenada originalmente en los bits menos significativos es aleatoria, con lo que su modificación para introducir información oculta no desvela que la imagen está tratada. Además la información introducida usando el sistema LSB es altamente vulnerable y podría ser destruida aplicando una ligera modificación a la imagen portadora como es la compresión que impone el formato JPEG (*Join Picture Experts Group*) [9].

SISTEMAS DE INSERCIÓN

Esta técnica trata de añadir los bits de información a partir de un determinado *token* o marca estructural del archivo. Esto puede ser en el fin del fichero (**EOF**), en los espacios de *padding* o alineamiento, metadatos, y otras estructuras presentes en el documento.

Es un método de ocultación trivial y además no modifica la apariencia de la imagen. Sin embargo, el mayor inconveniente que presenta es que sí modifica el tamaño del objeto contenedor, en este caso la imagen, con lo cual puede levantar sospechas si la cantidad de información introducida es considerablemente grande.

TÉCNICAS EN EL DOMINIO DE LA TRANSFORMADA

Al contrario que las técnicas que hacen uso del dominio espacial para insertar la información (técnicas de sustitución como LSB), las transformaciones en el dominio de la frecuencia ocultan la información secreta en las partes significativas de la cubierta. De este modo, para destruir el

mensaje serían necesarias muchas modificaciones sobre la imagen portadora. Las técnicas frecuenciales se consideran más robustas a los ataques que las espaciales.

Existen muchas transformaciones usadas para mapear una señal en el dominio de la frecuencia, las más usadas son:

- Transformada discreta del coseno (**DCT**).
- Transformada discreta de Wavelet (**DWT**).
- Transformada discreta de Fourier (**DFT**).

Cuando se añade información secreta a algunas componentes del dominio frecuencial se modifica toda la imagen en vez de una parte concreta de la misma. Esto sucede porque las frecuencias están distribuidas por toda la imagen.

TÉCNICAS DE ESPECTRO DISPERSO

En esta técnica, el dominio frecuencial del archivo portador se utiliza como canal de comunicación, de manera que el mensaje secreto se transmite como una señal a través de él. Es considerada como una de las técnicas más robustas de esteganografía ya que el mensaje se reparte por una banda de frecuencia.

TÉCNICAS ESTADÍSTICAS

Las técnicas estadísticas ocultan un solo bit de la información secreta en la imagen. Si se oculta un “1” algunas características estadísticas del archivo (entropía, probabilidad de distribución, etc.) deben ser cambiadas significativamente para indicar la existencia de un mensaje. En cambio, si el bit ocultado es un “0”, la cubierta queda intacta. La técnica depende enteramente en la habilidad del receptor para diferenciar entre una imagen modificada y otra que está intacta [10].

TÉCNICAS DE DISTORSIÓN

La mayoría de las técnicas esteganográficas no requieren que el receptor posea la imagen original y la portadora para extraer el mensaje, y es suficiente con tener esta última. Sin embargo, cuando empleamos técnicas de distorsión, el receptor necesita el archivo de cubierta original para recuperar el mensaje. De este modo, basta con aplicar una diferencia entre el archivo portador recibido y la imagen original para obtener el mensaje oculto.

TÉCNICAS DE GENERACIÓN DE CUBIERTA

Todas las técnicas anteriormente explicadas requieren un archivo de cubierta como contenedor para introducir el mensaje. En este caso, se codifica la información de tal forma que se crea una cubierta de comunicación secreta.

2.1.5 ESTEGANOGRAFÍA Y CRIPTOGRAFÍA

La mayor parte del arte de la esteganografía se basa en la búsqueda de algoritmos que hagan que un conjunto de datos parezca otro distinto. En algunos casos, camuflar la existencia de datos no es suficiente. Los ataques más potentes contra la esteganografía requieren medidas más fuertes, una de esas medidas se basa en el empleo de la criptografía.

Teniendo unos conocimientos básicos de esteganografía y criptografía podemos ocultar nuestros datos con un grado de seguridad altísimo. Mediante técnicas esteganográficas podemos hacer que cualquier información pase inadvertida (mostrando información inofensiva), pero la seguridad intrínseca que aporta la esteganografía para los datos importantes no es mucha.

Mediante la combinación de estas dos técnicas podemos establecer dos capas en la seguridad de la información: la primera de ellas, la esteganografía; y la segunda, interna, la criptografía. Cada una de las capas tiene un cometido: la criptográfica se encarga de la seguridad de los datos, y la esteganográfica protege la integridad de la capa criptográfica.

Por otro lado, cuanto mayor sea la encriptación de los datos que queramos camuflar, mayor será la aleatoriedad de los mismos, lo cual nos es muy beneficioso para ocultar los datos entre el ruido de la imagen sin levantar sospechas.

La forma más versátil y útil de criptografía es aquella que emplea una clave en el algoritmo. La clave es una colección de bits que juegan un papel esencial en la fortaleza del algoritmo empleado. La mayoría de las técnicas con clave usadas en la esteganografía provienen de las soluciones usadas en criptografía. Algunos de los tipos básicos son:

- **Claves Secretas**: Una clave es usada para esconder la información, y la misma clave es utilizada para encontrarla. A este tipo de clave se la conoce popularmente como clave simétrica.
- **Claves públicas**: Una clave oculta la información y otra distinta la revela. Es conocido como clave asimétrica ya que hay 2 claves diferentes.

Uno de los algoritmos más usados desde 1970 ha sido el de encriptación DES. Actualmente ha sido sustituido por *Rijndael*, que es más seguro y rápido. El diseño básico de ambos algoritmos se basan en dos principios básicos: **confusión y difusión**. La confusión consiste en modificar un mensaje de una manera no lineal. En cambio, la difusión toma una parte del mensaje y modifica la otra parte, de manera que cada parte del mensaje final depende de otras muchas partes del mensaje.

En todo caso es importante escoger en estos casos una clave secreta con cierta complejidad para evitar posibles ataques de diccionario. Un punto esencial del cifrado con clave simétrica es tener cuidado con el canal de distribución de la clave secreta.

Por otro lado, la encriptación con clave pública es algo más compleja que la anterior. El algoritmo más popular es el RSA. Cada persona crea un par de llaves y publica una de ellas, la otra se guarda en secreto. Si alguien quiere mandarle un mensaje a la otra, tiene que buscar la

clave pública del destinatario y cifrar el mensaje con ella. Tras este punto sólo puede el receptor legítimo descifran usando su clave privada. El sistema es bastante seguro y fiable.

En conclusión podemos decir que la encriptación es también una de las bases de los algoritmos empleados en esteganografía. Los algoritmos que toman un bloque de información y lo ocultan en el ruido de una imagen necesitan datos que sean lo más aleatorios posibles. Sin embargo, nada es perfecto, y en algunas ocasiones los datos pueden ser demasiado aleatorios y no pasar desapercibidos a la vista de un programa de estegoanálisis. En el siguiente apartado se realiza una descripción de los protocolos esteganográficos, en los que se describen con más detalle los sistemas de clave pública y privada.

2.1.6 PROTOCOLOS ESTEGANOGRÁFICOS

En el arte de la esteganografía existen principalmente 3 protocolos diferentes a través de los cuales se produce la comunicación oculta entre el emisor y receptor del mensaje. La diferencia esencial entre dichos protocolos se produce en el proceso de ocultación del mensaje y en el de extracción. Los protocolos disponibles para la esteganografía son:

ESTEGANOGRAFÍA PURA

En la esteganografía pura no se necesita un previo intercambio de información (como una clave) entre emisor y receptor. Este sistema únicamente requiere que ambos tengan acceso al algoritmo de ocultamiento y extracción. Esta estrategia basa la **seguridad en la oscuridad**, y es necesario mantener a salvo el algoritmo para que las comunicaciones no sean descubiertas fácilmente.

ESTEGANOGRAFÍA DE CLAVE PRIVADA

Con la esteganografía pura no es necesario ningún tipo de intercambio de información para comenzar el proceso de comunicación. Para aumentar la seguridad del sistema anterior, se va a emplear una llave secreta para cifrar el mensaje antes de ser ocultado, de modo que el receptor debe conocer la clave simétrica usada por el emisor para poder recuperar el mensaje secreto. El único inconveniente del sistema es que ambos participantes en la comunicación tienen que conocer la clave de cifrado previamente o intercambiarla por un canal seguro.

ESTEGANOGRAFÍA DE CLAVE PÚBLICA

Al igual que en la criptografía de clave pública, la esteganografía de clave pública no confía en el intercambio de una clave secreta. En este tipo de sistemas se requieren dos llaves, una pública y otra privada. La clave pública se usa durante el proceso de ocultamiento de la información, y la privada en el proceso de extracción del mensaje. En este caso también es necesario que los participantes en la comunicación privada conozcan la clave pública del receptor.

2.1.7 ESTEGOANÁLISIS

El estegoanálisis es la ciencia y arte encargada de la detección de mensajes ocultos en medios digitales, como son las imágenes. Se podría comparar con el criptoanálisis, que se encarga de analizar e intentar romper criptosistemas, sin embargo, el objetivo principal del estegoanálisis no es obtener el mensaje oculto, sino confirmar la existencia de información secreta en el medio. La recuperación de dicha información queda fuera de los límites del estegoanálisis, en la mayoría de casos resultaría imposible obtener el secreto oculto, pues las técnicas esteganográficas cifran la información utilizando criptografía y dispersan la información pseudo-aleatoriamente por el medio.

La simple detección de la existencia de mensajes ocultos no es una tarea trivial, requiere de análisis profundo del medio. Las técnicas de detección más empleadas en este arte se clasifican dependiendo del sistema de análisis empleado:

ESTEGOANÁLISIS MANUAL

Se trata de buscar manualmente diferencias entre el objeto contenedor (original) y el portador del mensaje, de manera que encontremos cambios en la estructura que nos indiquen la presencia de datos ocultos. El problema de esta técnica es que necesitamos disponer del objeto original y del portador, y aún así hay ocasiones en las que no podemos extraer el mensaje.

Sin embargo como hemos comentado, una vez dispongamos de ambos ficheros podemos buscar evidencias en el fichero esteganografiado que muestren indicios de existencia de información secreta.

Uno de los posibles ataques que encontramos en este campo es el **ataque visual**, que alerta al ojo humano de la presencia de información oculta gracias a la aplicación de filtros. En caso de no disponer del fichero original se pueden buscar irregularidades pero, como ya hemos dicho anteriormente, difícilmente podremos obtener información útil más allá de la existencia de los mismos.

ESTEGOANÁLISIS ESTADÍSTICO

Se realiza un análisis estadístico de las frecuencias de aparición y distribución de los colores de la imagen portadora. Los programas que realizan este tipo de análisis tratan de buscar patrones comunes empleados en programas de esteganografía para detectar la presencia de un mensaje oculto.

Si bien, tenemos que comentar que la inserción manual de mensajes ocultos puede burlar este tipo de análisis. El estegoanálisis estadístico suele dar mejores resultados que el manual, sin embargo es una técnica que necesita mucho tiempo para procesar la imagen completamente.

Existen técnicas de ataque de este tipo, una de las más empleadas es el ataque ***Chi-Square***, que permite estimar el tamaño de la posible información oculta en una imagen [8].

2.1.8 ATAQUES A LA ESTEGANOGRAFÍA

Algunos de los ataques más comunes a la esteganografía en imágenes se describen brevemente a continuación:

- **Ataque a un fichero**: El atacante tiene acceso a un fichero y debe determinar si contiene un mensaje oculto. Es el tipo de ataque más débil, de hecho se considera más un arte que una ciencia. Este ataque se basa en el análisis estadístico y visual del archivo.
- **Ataque con fichero estenografiado y original**: Si el atacante tiene acceso a una copia del fichero original y al portador del mensaje puede realizar ataques de comparación para certificar si existe un mensaje oculto. Una vez comprobado, el atacante puede optar entre: intentar eliminar el mensaje oculto; cambiar el mensaje oculto por uno propio; o intentar extraer el mensaje oculto original.
- **Múltiples archivos codificados**: El atacante obtiene n copias diferentes de un archivo con n mensajes distintos. Puede intentar destruir todos los mensajes ocultos o mezclar todos los archivos juntos.
- **Acceso al fichero y al algoritmo**: Un algoritmo esteganográfico ideal es aquel que a pesar de que el atacante lo posea, le sea imposible recuperar el mensaje original ocultado. Evidentemente esto sólo ocurre con aquellos algoritmos basados en la criptografía, donde se requiere una clave para extraer los datos.
- **Ataque de destrucción total**: El atacante puede destruir el mensaje fácilmente difuminando la imagen portadora o añadiendo ruido a la misma. Una de las técnicas más comunes consiste en rotar la imagen 45 grados, difuminar la imagen, volver a darle nitidez, y rotarla de nuevo, con esto habríamos conseguido destruir el mensaje.
- **Eliminar bits menos significativos**: Si cierta imagen lleva un mensaje ocultado usando la técnica LSB es susceptible de recibir este ataque, que se encargaría de cambiar el valor de los bits menos significativos de la imagen para borrar el mensaje portado.
- **Ataque de adición de información**: Si el atacante usa el mismo software para incrustar un nuevo mensaje en la imagen puede llegar a sobrescribir el mensaje original con el suyo. Obviamente esto sólo ocurre con ciertos algoritmos. Además se puede evitar introduciendo códigos de corrección de errores.
- **Ataque de cambio de formato**: Cuando cambiamos el formato de cierta imagen podemos perder los datos que estaban ocultos en la misma. Esto ocurre porque no todos los formatos guardan la información de la misma manera.
- **Ataque de compresión**: Se trata de uno de los ataques más sencillos de todos. Los algoritmos de compresión intentan eliminar la información extra que encuentran en los ficheros. Frecuentemente la información oculta coincide con la información extra que es eliminada. Los algoritmos de compresión agresivos no suelen reconstruir el fichero original tras descomprimirlo.

2.2 IMÁGENES DIGITALES

La esteganografía puede emplear prácticamente la mayoría de medios digitales para ocultar información. Uno de los medios más populares dentro de la ciencia esteganográfica es la imagen digital.

Una imagen digital no es más que una representación en dos dimensiones de una matriz numérica, que comúnmente está codificada en binario. Se distinguen dos tipos de imágenes digitales: las matriciales o **mapas de bits** y las vectoriales. La diferencia elemental radica en que las imágenes vectoriales están formadas a partir de objetos geométricos y polígonos, en cambio los mapas de bits están formados por estructuras más heterogéneas.

Los mapas de bits son las imágenes más utilizadas en las técnicas de ocultación, en el siguiente apartado se van a introducir los principales modelos de color que existen.

2.2.1 MODELOS DE COLOR

Las imágenes se representan en forma de píxeles. Dependiendo del número de colores, las imágenes se pueden diferenciar en diferentes tipos [11]:

IMÁGENES BINARIAS

Las imágenes binarias sólo tienen dos colores, el blanco y el negro. Cada píxel que conforma la imagen puede tener dos valores, un “1” representa el blanco y “0” el negro [12]. La utilidad de este modelo de color se encuentra en el campo del procesamiento y segmentación de imágenes, pero en lo que concierne a la esteganografía no es muy empleado.

IMÁGENES EN ESCALA DE GRISES

En una imagen con escala de grises, los píxeles pueden ser blancos, negros o tonos grisáceos intermedios [12]. Cada píxel está formado por 8 bits. De manera que poseemos 256 posibilidades de color en la escala. Por ejemplo un píxel que contuviera “00000000” representaría al color negro, otro con el valor “11111111” equivaldría al blanco, y los bits “00110101” conformarían un gris oscuro [11]. En la figura 5 se representa la paleta de colores para la escala de grises.

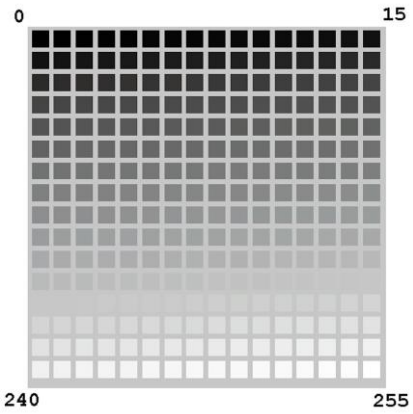


Ilustración 5: Paleta de escala de grises.

MODELO DE COLOR RGB

En las imágenes que emplean el modelo de color RGB, cada píxel contiene 24 bits. Esos 24 bits se reparten en 3 bytes, los primeros 8 bits conforman el color rojo, los 8 siguientes el verde, y los 8 últimos bits colorean el azul. Por tanto se tienen 256 posibles tonalidades para cada uno de los 3 colores básicos. Combinando las tres componentes se pueden representar hasta 2^{24} colores distintos.

MODELO DE COLOR CMYK

Las imágenes se construyen usando la mezcla de cuatro colores y se representan con un porcentaje entre 0 y 100%. CMYK representa cian, magenta, amarillo, y negro. Siempre que se necesita imprimir una imagen se emplea el modelo CMYK en lugar del RGB, ya que representa con mayor fidelidad la pigmentación del color [12].

MODELO DE COLOR HSV

Este modelo se obtiene a través de una transformación no lineal del espacio de color RGB, y se emplea para representar progresiones de color. Para definir un color se utilizan sus tres componentes: matiz, saturación y brillo.

CODIFICACIÓN YCBCR

Se trata de una familia de espacios de colores que se obtiene al codificar la información RGB. Las componentes que emplea son: la luminancia (Y), la componente de color azul (Cb), y la componente de color rojo (Cr). Esta codificación tiene gran importancia ya que es empleada por el formato de imagen JPEG, que como dijimos anteriormente es el más empleado en el mundo esteganográfico.

2.2.2 FORMATOS DE IMAGEN

Los formatos de imagen más populares son:

WINDOWS BITMAP (BMP)

Es el formato de gráfico más simple, casi nunca en la práctica usa compresión. Consta de una cabecera y a continuación los valores de cada píxel de la imagen. Su principal ventaja es la sencillez (es el formato perfecto para realizar esteganografía). El único problema que presenta es el enorme espacio que ocupa.

PC PAINTBRUSH (PCX)

Es la evolución del mapa de bits tradicional. Se usa el algoritmo de compresión RLE (cuando dos o más píxeles consecutivos tienen el mismo color, el algoritmo guarda la información del color y el número de píxeles que lo usan). La principal ventaja es la sencillez del algoritmo empleado, y el inconveniente es la escasa compresión obtenida finalmente.

GRAPHICS IMAGE FORMAT (GIF)

Se caracteriza por ser uno de los mejores formatos de compresión existentes. Usa el algoritmo de compresión LZW, mucho más complejo que RLE. La desventaja que encontramos en este formato es la limitación de los colores (256 colores posibles). La ventaja es la enorme compresión obtenida y la capacidad de uso de transparencias y entrelazado.

JOINT PHOTOGRAPHIC EXPERTS GROUP (JPEG)

Es el tipo de fichero más popular. El algoritmo empleado para la compresión se basa en un defecto del ojo humano que impide la completa visualización de la paleta de 24 bits, por lo que se elimina la información que el ojo humano es incapaz de procesar. Este algoritmo sí tiene una importante pérdida de información en el proceso de compresión (*loosy*). Por lo tanto, la principal ventaja es la calidad de la imagen en poco espacio, y la desventaja es la pérdida de información. En la esteganografía es el tipo de fichero más usado, sin embargo no es el más sencillo precisamente. En el presente proyecto tendremos que lidiar con este formato y sortear el proceso de compresión que impone.

TAGGED IMAGE FILE FORMAT (TIFF)

Es un formato de altísima resolución y calidad. Usa el estándar CMYK en el mapa de bits. No es muy empleado en la esteganografía dado su alto tamaño.

PORTABLE NETWORK GRAPHICS (PNG)

Es otro de los formatos de imagen más interesantes para emplear con técnicas esteganográficas ya que cubre todas las características del formato GIF pero con un mejor algoritmo de compresión y sin pérdida de información. Estamos hablando de una paleta de color muy superior a los 256 bits del formato GIF (el doble). Este tipo de fichero contiene información adicional en forma de metadatos, que nos otorga nuevas posibilidades de ocultación de información. También manejaremos este formato de imagen en el proyecto.

2.2.3 PLANOS DE BITS

Hemos visto que las imágenes digitales se representan en forma de píxeles que, a su vez, dependen de los bits que los conforman. Una imagen se puede dividir en diferentes planos de bits [12]. Por ejemplo, si tenemos una imagen en escala de grises, donde cada píxel se representa con 8 bits (como se ha explicado en la sección de modelos de color), se pueden obtener hasta 8 planos de bits diferentes. El plano 0 contiene los bits menos significativos de la imagen, el plano 7 contiene los datos visualmente más importantes, y el resto de planos contribuyen en mayor o menos detalle a la imagen (ver figura 6). Los planos de bits son muy empleados en el procesamiento de imagen y en las herramientas de estegoanálisis. Técnicas como la LSB pueden ser detectadas fácilmente analizando los planos de bits menos significativos en una imagen. Uniendo todos los planos se recompone la imagen original.

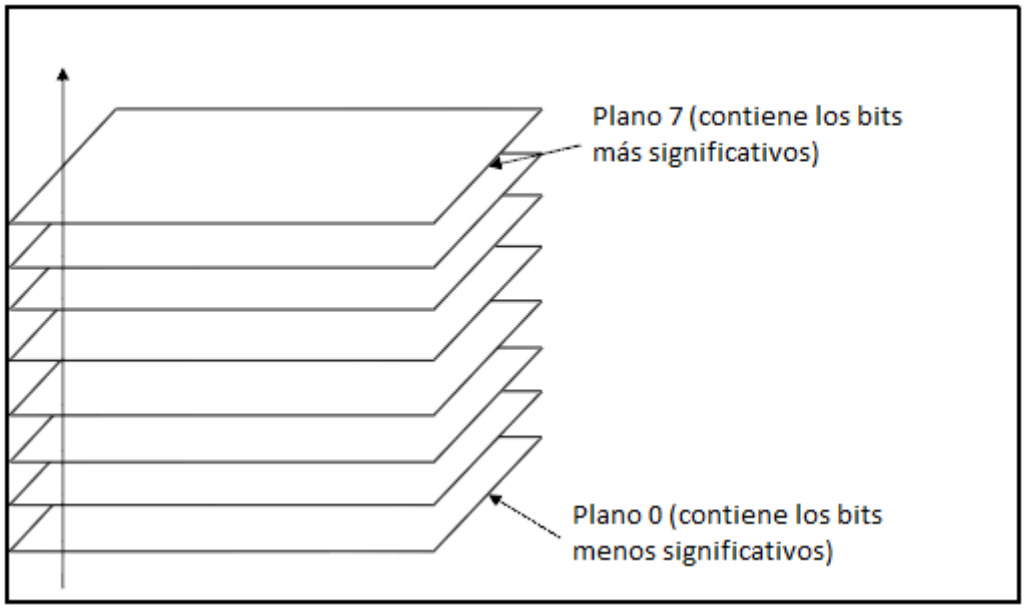


Ilustración 6: Planos de bits en una imagen en escala de grises [13]

2.2.4 COMPRESIÓN

La compresión en el contexto de las imágenes digitales es una práctica muy habitual, que reporta múltiples beneficios en muchos casos. El objetivo principal de la compresión es reducir la información redundante e irrelevante que contenga la imagen para permitir su almacenamiento o transmisión de manera eficiente [14].

Existen dos tipos de compresión: la que introduce pérdidas o también llamada *loosy*, y la compresión sin pérdida. Mientras que esta última se utiliza para el almacenamiento de imágenes de alta resolución como fotografías, la compresión con pérdidas es ampliamente utilizada en el contexto de Internet y las comunicaciones.

La compresión de imágenes también es utilizada en la esteganografía diariamente. En este contexto, la compresión sin pérdidas ofrece más posibilidades, ya que mantiene los datos exactamente como estaban originalmente. Por otro lado la que introduce pérdidas podría hacer perder la integridad original de la imagen [15], lo que se traduce en la incapacidad de poder recuperar un posible mensaje oculto cuando se han empleado técnicas de sustitución como LSB.

COMPRESIÓN JPEG

Un ejemplo de formato que impone compresión con pérdidas es JPEG, en el que la imagen reduce notablemente su tamaño pero sin reducir notoriamente la calidad para el ojo humano. Siendo más concretos, JPEG extrae toda la información que el ojo humano no es capaz de ver y la elimina, los procesos que se realizan durante la compresión en este formato son [16]:

1. Se convierte el valor de los píxeles al modelo de color YCbCr.

2. Se reducen los valores de crominancia (el ojo es más sensible a cambios de luminancia que a variaciones significativas de color).
3. Se transforman los valores al dominio frecuencial (utilizando la DCT).
4. Se contabilizan los valores obtenidos y se dividen individualmente entre unos valores predeterminados, finalmente se redondean los resultados al número real más cercano.
5. Se realiza el ordenamiento en "Zigzag" de los valores cuantificados.
6. Compresión final sin pérdidas.

Durante la etapa de "cuantización" (proceso 4) se produce la pérdida de información, concretamente cuando se rodean los coeficientes al valor real más cercano. El resto de etapas se consideran reversibles. Cuanto mayor sea el factor de compresión menor será la calidad final de la imagen resultante. La figura 7 muestra el proceso de compresión de JPEG.

Uno de los principales objetivos del presente proyecto consistirá en sortear la compresión que impone JPEG y evitar que la información que ocultemos en las imágenes se pierda. Por lo que será necesario emplear los bits más significativos en vez de usar técnicas como la LSB.

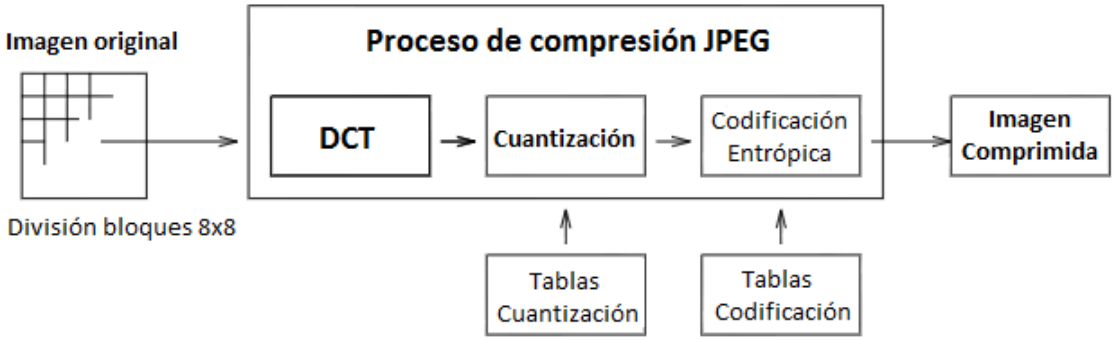


Ilustración 7: Proceso de compresión JPEG

2.3 ANDROID

2.3.1 INTRODUCCIÓN A LOS *SMARTPHONES*

Los teléfonos inteligentes o *smartphones* son teléfonos móviles, pero con características más avanzadas que éstos y, similares a las de un ordenador. Algunos de ellos son táctiles y tienen conexión a Internet; existen múltiples tecnologías de conexión a Internet desde estos dispositivos como son 3G, Wi-Fi, WAP, GPRS, EDGE, EvDO, HSD, HSDPA, etc. [3].

Los smartphones han permitido a los usuarios poner a disposición de éstos, información en tiempo real, en todo tipo de lugares y situaciones. El desarrollo de estas tecnologías para teléfonos inteligentes, ha ido de la mano de las redes sociales y de las aplicaciones de mensajería instantánea, que han cambiado la forma de comunicarnos en el mundo. Esta comunicación se produce en tiempo real, lo cual permite que, independientemente del desarrollador o del sistema operativo, los usuarios se interconecten a través de sus terminales móviles.

2.3.2 SISTEMA OPERATIVO ANDROID

Android es una plataforma para dispositivos móviles, su sistema operativo está basado en **Linux**, del cual dependen los servicios base del sistema como pueden ser la seguridad, gestión de memoria, gestión de procesos, controladores, etc. [3]. Su código fuente está disponible bajo licencias de software libre y código abierto.

El sistema operativo Android, propiedad de Google, tuvo un desarrollo plenamente enfocado a Internet, de ahí que sus fabricantes integraran todo tipo de aplicaciones relacionadas con las redes sociales y mensajería. El núcleo del sistema operativo está programado en lenguaje C (más de 2.8 millones de líneas de código), algunas librerías también están en C++ (1.7 millones de líneas), sin embargo, la interfaz principal y las aplicaciones son programadas en lenguaje **JAVA** (2.1 millones de líneas).

El hecho de que las aplicaciones sean desarrolladas en JAVA ha atraído a cientos de programadores a esta plataforma, debido a su flexibilidad, facilidad y usabilidad. Es relativamente sencillo crear aplicaciones para este sistema, de hecho es una de las razones por las que se ha escogido este sistema operativo sobre sus competidores como iOS u otros.

Actualmente Android es el dominador del mercado de sistemas operativos para móviles con una cuota de más del 36%. Se encuentran disponibles más de 400 mil aplicaciones compatibles con este sistema [17].

2.3.3 COMUNICACIONES MULTIMEDIA

Las comunicaciones multimedia son uno de los puntos fuertes de los teléfonos inteligentes de última generación. Una comunicación multimedia es aquella en la que se intercambian medios digitales entre los usuarios, como pueden ser imágenes, sonidos, vídeos, etc. Es la evolución natural de la mensajería de texto tradicional.

Actualmente las comunicaciones multimedia en dispositivos móviles se realizan principalmente a través de 3 sistemas distintos:

- **Correo electrónico**: La mayoría de dispositivos móviles se encuentras conectados y sincronizados continuamente con el correo personal de los propietarios. El correo electrónico permite el envío de elementos multimedia adjuntos a los mensajes.
- **Sistema de mensajería multimedia (MMS)**: Es un estándar de mensajería que permite el envío y recepción de contenido multimedia entre distintos terminales. El MMS también permite el envío a correos electrónicos particulares. Sin embargo, el principal inconveniente que posee, aparte de ser un servicio de pago, es que el tamaño del archivo multimedia intercambiado no puede superar (en la mayoría de los casos) 300KB, lo cual limita las comunicaciones.
- **Sistemas de mensajería instantánea (IM)**: Sin duda el sistema de comunicación para móviles más utilizado en la actualidad (excluyendo a las redes sociales). Se trata de una comunicación en tiempo real entre dos o más personas conectadas a una red como Internet. En un principio sólo era posible enviar mensajes de texto, sin embargo se han ido adaptando a los nuevos tiempos y ahora es posible intercambiar medios digitales. Todo ello totalmente gratuito, y sólo se necesita conexión a Internet. La aplicación más popular a día de hoy para dispositivos móviles se llama ***WhatsApp***.

En los 3 sistemas de comunicaciones explicados anteriormente es posible realizar el envío de imágenes, algunos de ellos imponen compresión sobre las mismas antes de ser enviadas (como es el MMS y la mensajería instantánea) y otros no (como el correo electrónico). En el presente trabajo se intentará desarrollar una aplicación esteganográfica que sea capaz de enviar imágenes con mensajes ocultos a través de alguno de estos sistemas.

2.4 CONTEXTO ACTUAL

Como se ha comentado en la introducción del trabajo, la esteganografía nos brinda la oportunidad de comunicarnos secretamente sin levantar sospechas. Esto se ha hecho durante siglos, la principal diferencia es que antiguamente el canal de comunicación era secreto y ahora se usan canales públicos y no seguros como Internet.

En los últimos tiempos la esteganografía ha sido utilizada con fines muy dispares, es bien conocido que los atentados terroristas del 11 de Septiembre se planearon utilizando la esteganografía sobre imágenes subidas a sitios web públicos [6]. Pero una vez más he de decir que los medios no son culpables de lo que haga la gente con ellos. En contraposición, la esteganografía ayuda, entre muchas cosas, a mantener comunicaciones ocultas entre agencias militares y de inteligencia, policías infiltrados en redes criminales, mantener el anonimato en el voto electrónico [5], en la lucha de los civiles contra las restricciones impuestas por el Estado, etc.

Existen múltiples aplicaciones para la mayoría de plataformas que permiten la ocultación de mensajes en imágenes, algunos de los más conocidos son: *F5, Jsteg, OutGuess, EzStego* y *Steganos* [5]. La mayoría de la esteganografía moderna sobre imágenes trabaja en el dominio frecuencial (DCT, Wavelet, etc.), y las imágenes obtenidas se encuentran en formato JPEG.

Para dispositivos móviles existen también aplicaciones similares, en el caso de Android las que tienen mayor relevancia son: *MobiStego, Secret Letter* y *StegDroid Alpha* [18]. Son aplicaciones sencillas que permiten el envío de imágenes esteganografiadas por MMS y correo electrónico.

El problema nos alcanza cuando necesitamos transferir una imagen a través de un medio que impone una compresión agresiva, esto sucede al enviar una imagen usando aplicaciones de mensajería instantánea, en estos casos las herramientas anteriormente citadas fallan y el mensaje oculto se pierde o queda dañado.

Se abre, por tanto, la necesidad de desarrollar una aplicación que nos permita enviar mensajes ocultos a través de este tipo de medios, que son los que más usa la gente. Como no todos las aplicaciones de mensajería imponen la misma compresión, se tomará como referencia WhatsApp, que es la aplicación más popular para Android.

3 MARCO REGULADOR

Se ha realizado un estudio de las normativas técnicas y leyes vigentes que afectan al marco legal del proyecto. A continuación se explican detalladamente cada una de ellas:

3.1 LEY ORGÁNICA DE PROTECCIÓN DE DATOS (LOPD)

La aplicación a desarrollar en este proyecto va a trabajar con información sensible del usuario, como son los mensajes secretos, sus contraseñas, y las propias imágenes portadoras que éste escoja. Por lo tanto, es necesario contemplar la ley de Protección de Datos de Carácter Personal.

El objeto de la presente Ley Orgánica *es **garantizar y proteger**, en lo que concierne al tratamiento de los **datos personales**, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor personal y familiar* [19].

En otras palabras, su objetivo principal es regular el tratamiento de los datos y ficheros, de carácter personal, independientemente del soporte en el cual sean tratados, los derechos de los ciudadanos sobre ellos y las obligaciones de aquellos que los crean o tratan [20].

La Ley Orgánica 15/1999, del 13 de Diciembre, de Protección de Datos de Carácter Personal expone en su artículo 9, sobre la seguridad de los datos, que: "*El responsable del fichero, y, en su caso, el encargado del tratamiento deberán adoptar las medidas de índole técnica y organizativas necesarias que garanticen la seguridad de los datos de carácter personal y eviten su alteración, pérdida, tratamiento o acceso no autorizado, habida cuenta del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que están expuestos, ya provengan de la acción humana o del medio físico o natural*."

En lo que concierne a la aplicación, se han tomado las siguientes medidas:

- Ningún servidor o sistema perteneciente a la aplicación almacenará datos sobre el usuario. El sistema fue concebido con el fin de proteger y ocultar la información. Las imágenes generadas serán enviadas a la aplicación de mensajería que el usuario escoja bajo su responsabilidad.
- Únicamente el autor legítimo del mensaje secreto y los participantes que él considere de autoría (receptores) serán capaces de acceder a la información secreta, conociendo la contraseña de ocultación y teniendo acceso a la imagen portadora.
- Se han empleado técnicas criptográficas y esteganográficas para proteger los datos manejados por la aplicación. El algoritmo de cifrado empleado sobre el mensaje presume de ser uno de los más seguros, de modo que ningún ataque contra la seguridad propone el sistema será viable.

3.2 LEY GENERAL DE LAS TELECOMUNICACIONES

La segunda ley que afecta al sistema es la Ley General de las Telecomunicaciones, cuyo objetivo es **garantizar el secreto de las comunicaciones**. Concretamente el artículo 36 de la Ley General de Telecomunicaciones, del 3 de Noviembre de 2003, sobre el "Cifrado en las redes y servicios de comunicaciones electrónicas" dicta [21]:

> *1. Cualquier tipo de información que se transmita por redes de comunicaciones electrónicas podrá ser protegida mediante procedimientos de cifrado.*
>
> *2. El cifrado es un instrumento de seguridad de la información. Entre sus condiciones de uso, cuando se utilice para proteger la confidencialidad de la información, se podrá imponer la obligación de facilitar a un órgano de la Administración General del Estado o a un organismo público, los algoritmos o cualquier procedimiento de cifrado utilizado, así como la obligación de facilitar sin coste alguno los aparatos de cifra a efectos de su control de acuerdo con la normativa vigente.*

La aplicación desarrollada emplea técnicas de cifrado sobre los datos transmitidos dentro del medio digital portador, que en nuestro caso son las imágenes. Se puede decir, por tanto, que se garantiza el secreto de las comunicaciones, y más aún cuando las mismas son ocultadas empleando esteganografía.

En el caso de que la Administración General del Estado o un organismo público requieran los algoritmos o procedimientos de cifrado empleados, se adjunta un correo electrónico de contacto tanto en la aplicación como en la página de descarga en *Google Play*.

3.3 LEY DE LA PROPIEDAD INTELECTUAL

El proyecto también se basa en la presente ley, ya que uno de los usos más frecuentes de la esteganografía y, más concretamente, de las técnicas de marca de agua o *watermarking*, es **preservar la autoría** de cualquier tipo de soporte tangible o no. Entre los tipos de objetos de creación individual que recoge dicha ley están las fotografías, imágenes fijas, e imágenes tomadas por exposición, que son los medios empleados en la aplicación que presentaremos en el siguiente capítulo.

Le ley de Propiedad Intelectual, del 22 abril de 1996, en su primer punto del artículo 6 dicta que [22]:

- *La presunción de autoría puede ser demostrada por afirmación o por firma, en otro caso tan solo se podrá afirmar que es de alguien demostrándolo.*

En nuestro caso, la autoría de la obra esteganográfica puede ser demostrada mediante el conocimiento del autor legítimo de la clave de cifrado y ocultación empleada. De hecho, como se comentó en los posibles usos de la esteganografía, puede ser empleada como medio de autenticación demostrando el conocimiento de la clave.

4 ANÁLISIS

El objetivo que persigue el análisis del sistema de información es obtener una especificación detallada del mismo. Esta especificación conformará la base para el posterior diseño de la aplicación. El análisis, por tanto, no pretende solucionar el problema, que será el cometido del diseño, sino tratar de captar las necesidades que se deben resolver y modelar el problema [23].

4.1 DEFINICIÓN DEL SISTEMA

El primer paso a la hora de analizar un sistema es detallar su funcionamiento, los problemas que debe resolver, las restricciones que va a tener y el entorno operacional que requerirá. Es necesario conocer estos detalles antes de adentrarse en la captación y análisis de requisitos.

4.1.1 ALCANCE DEL SISTEMA

El sistema de información que se pretende diseñar consiste en una aplicación que va a permitir a los usuarios de dispositivos móviles Android mantener comunicaciones seguras y ocultas por medios como los sistemas de mensajería instantánea, entre otros. En otras palabras, podría calificarse como “complemento” de seguridad y privacidad para las aplicaciones que permiten intercambiar mensajes entre los usuarios.

La aplicación debe ser capaz de ocultar un mensaje secreto dentro de un fichero aparentemente inofensivo como es una imagen. Posteriormente debe adaptar dicha imagen portadora al medio por el que va a ser transmitida, en este caso por las redes de telefonía e Internet que emplean los programas de mensajería móvil. Todo ello para que finalmente sea recibida por el destinatario o destinatarios y pueda recuperar el mensaje original que se pretendía comunicar.

Es necesario que el programa implemente algoritmos esteganográficos para la ocultación y criptográficos para el cifrado. Asimismo el mensaje secreto debería ser recuperable a pesar de sufrir modificaciones si es comprimido en su camino al receptor, para ello debe contar con mecanismos de recuperación ante errores.

La funcionalidad completa se definirá a lo largo del presente documento. En el siguiente apartado se identificarán las restricciones que afectan al presente sistema de información.

4.1.2 RESTRICCIONES GENERALES

Las siguientes restricciones han sido impuestas por el equipo de desarrollo y por la normativa legal comentada en el marco regulador (sección 3) para el actual proyecto:

- La interfaz del programa debe ser amigable, usable y lo más sencilla posible.
- La aplicación debe contener todas las funcionalidades que aparecen en los requisitos de usuario.
- Los requisitos de usuario han sido recogidos a partir de las reuniones mantenidas con el tutor y de las decisiones que el alumno ha considerado más oportunas.

- El sistema ha de estar programado en JAVA y debe ser compatible con la mayoría de versiones del sistema operativo Android.
- Los algoritmos criptográficos que se empleen sobre el sistema no deben considerarse rotos (tienen que ser seguros).
- No se enviará información personal de los usuarios ni de los datos tratados por el programa a ningún servidor externo.
- Se cuenta con un presupuesto muy reducido, así que se escogerán herramientas gratuitas de diseño y desarrollo para la implementación del proyecto.

4.1.3 ENTORNO OPERACIONAL

El software desarrollado requiere unos requisitos mínimos para garantizar su correcto funcionamiento. El entorno operacional con el que debe contar el usuario para poder ejecutar la aplicación es el siguiente:

- Un dispositivo con una versión de Android igual o superior a la **2.2**, las versiones anteriores no cuentan con una API que permita el desarrollo de las funciones requeridas por el programa.
- Un terminal que disponga de **memoria RAM** suficiente para albergar las estructuras de datos con las que trabaja la aplicación. La mayoría de dispositivos físicos ya cuentan con este requerimiento, sin embargo aún existen algunos (sobre todo los más antiguos) cuyos mecanismos de manejo de memoria no soportan tanta carga computacional como la exigida.
- Ficheros de imágenes sobre los que pueda trabajar el programa.

Por otro lado, el entorno de desarrollo necesario para poder realizar la implementación del sistema va a ser descrita en el siguiente apartado.

4.2 ENTORNO DE DESARROLLO

Para llevar a cabo el presente proyecto ha sido necesario contar con un completo entorno de desarrollo, el cual está compuesto de los siguientes elementos hardware y software:

4.2.1 EQUIPOS

Los equipos y dispositivos físicos empleados son:

- ***Ordenador portátil Acer***
  - Modelo: Aspire 5930G.
  - Procesador: 2.26GHz Intel Centrino 2.
  - Sistema operativo: ArchLinux 3.7.5.
  - Memoria RAM: 4GB DDR2.
  - Disco duro: 320GB.
- ***Dispositivo móvil Samsung Galaxy***
  - Modelo: Galaxy S1 (GT-I9000).
  - Versión Android: 2.3.6 Gingerbread, nivel de API 10.
  - Procesador: 1Ghz-1.2GHz.
  - Memoria RAM: 353MB.
  - Kernel: Semaphore 2.4.0bm.

4.2.2 LENGUAJE DE PROGRAMACIÓN

Durante el desarrollo del sistema de información ha sido necesario emplear los siguientes lenguajes de programación:

- ***JAVA:*** Es el lenguaje de programación empleado a nivel de aplicaciones para los sistemas Android. Dependiendo de la versión de Android se dispone de una u otra API, en este caso la versión de ésta es la 10, que es la que tiene las interfaces de programación mínimas requeridas para implementar este sistema.
- ***Bash scripting:*** Es el lenguaje interpretado en la consola de comandos de los sistemas Linux. Además es compatible con POSIX. Este "lenguaje" se ha utilizado durante el proyecto en el manejo del sistema operativo Linux para ejecutar programas, realizar búsquedas, editar ficheros, etc.

4.2.3 ENTORNO DE DESARROLLO INTEGRADO (IDE)

Para desarrollar la aplicación se ha empleado ***Eclipse***, un entorno de desarrollo muy maduro programado en JAVA con gran cantidad de *plugins* y complementos para el diseño de aplicaciones. La información sobre su versión, entornos de desarrollo de software (SDK, "*Software Development Kit*") usados, y *plugins* instalados se resume a continuación:

- Versión: 3.4.
- Plugins: ADT (*Android Development Tools*), MAT (*Memory Analyser*).
- Android SDK: 2.2 (API 10).

Se ha escogido este IDE frente a otros como *Netbeans* por su nivel de integración para programar en lenguaje JAVA para dispositivos Android. Asimismo provee de un emulador virtual para Android que permite simular el funcionamiento de la aplicación en cualquier versión del sistema operativo y bajo circunstancias configurables.

4.2.4 SOFTWARE DE INGENIERÍA INVERSA

Se conoce por ingeniería inversa las técnicas y mecanismos que permiten obtener información a partir de un producto público. En nuestro caso, ha sido necesario realizar ingeniería inversa sobre las aplicaciones que se van a integrar con nuestro programa, con el fin de conocer sus detalles técnicos, funcionamiento interno, y parámetros de configuración.

Las herramientas que he usado para realizar ingeniería inversa sobre las aplicaciones de Android han sido:

- ***AppInstaller v.1.0.3****:* Aplicación para Android que permite exportar a formato APK (*Application PacKage*), que es un archivador de ficheros, las aplicaciones instaladas en Android. A partir de este APK es posible extraer su fichero ejecutable "*clases.dex*",
- ***Dex2Jar v.0.0.9.12:*** Convierte ficheros con la extensión DEX (*Dalvik EXecutable)*, que son empleados por la máquina virtual *Dalvik* de Android, a ficheros en formato CLASS de JAVA (ejecutables).
- ***JD-GUI v.0.3.5:*** Se trata de una aplicación que muestra el código fuente de los ficheros *class*, que hemos obtenido con *Dex2Jar.* Es un descompilador para JAVA. De esta forma podemos guardar los ficheros JAVA que contienen los métodos y parámetros que usa la aplicación para analizarlos con detenimiento.
- ***Shell de Linux:*** Con este intérprete de comandos tendremos a nuestra disposición el gran repertorio de herramientas que nos brinda el sistema operativo Linux. Lo emplearemos, entre otras cosas, para buscar cadenas de texto en los ficheros JAVA descompilados.
- ***GHex:*** Es un editor hexadecimal para el entorno gráfico *Gnome.* Con él podremos explorar el contenido de los ficheros de imágenes para analizar su cabecera y otros datos de interés.

4.2.5 SOFTWARE DE ESTEGOANÁLISIS

Se han empleado los siguientes programas para las pruebas de estegoanálisis:

- ***Stegdetect v.0.5****:* Una de las herramientas de estegoanálisis más famosas, es capaz de detectar información oculta en muchas imágenes esteganografiadas.
- ***XStegSecret v.0.1:*** Programa de estegoanálisis similar al anterior que analiza patrones estadísticos en imágenes para detectar la existencia de un mensaje oculto.
- ***StegSpy v.2.1:*** Analizador de imágenes esteganografiadas.
- ***PSNR Calculator v.1.5:*** Calculadora para obtener la "relación señal a ruido de pico" (PSNR) en imágenes.
- ***IM Processa v.1.6:*** Herramienta codificada en JAVA que permite obtener los diferentes planos de bits que componen una imagen.

4.3 REQUISITOS DE USUARIO

Los requisitos de usuario son los que recogen información sobre qué es lo que quiere el cliente y qué es lo que necesita. En nuestro caso, tratándose de un proyecto de fin de grado, la obtención de estos requisitos se ha realizado manteniendo sesiones de trabajo con el tutor del proyecto. En dichas sesiones se han definido las tareas que el sistema de información debería ser capaz de realizar, y las limitaciones que éste debe tener.

La especificación de requisitos de usuario se divide en **requisitos de capacidad** del sistema y **requisitos de restricción** impuestos sobre el sistema.

Cada requisito de usuario debe ser definido en una tabla con los siguientes atributos:

- **Identificación:** Cada requisito de usuario incluirá un identificador unívoco que facilitará su posterior trazabilidad. Estará compuesto de dos siglas, que se refieren a los requisitos de usuario (RU) de capacidad y a los de restricción, y dos números, que se emplearán para numerarlos.
- **Nombre:** Nombre general del requisito de usuario.
- **Descripción:** Breve descripción de la funcionalidad o restricción del requisito.
- **Prioridad:** Refleja la prioridad establecida para un requisito en el proceso de diseño o implementación. Los posibles valores que puede recibir este atributo son:
  - *Alta:* Se trata de un requisito que debe ser añadido antes que los de menor prioridad.
  - *Media:* Tras completar los requisitos de prioridad alta se implementan éstos antes que los que poseen prioridad baja.
  - *Baja:* Son los últimos en añadirse.
- **Necesidad:** Definen la importancia del requisito. Los valores que puede tomar este campo son:
  - *Esencial:* El requisito no será negociable y debe ser implementado.
  - *Deseable:* Es importante implementar dicho requisito pero no obligatorio.
  - *Opcional:* Se podrá implementar si las circunstancias lo permiten.
- **Fuente:** Indica el origen del requisito de usuario, en nuestro caso las fuentes son:
  - *Tutor:* El tutor del proyecto propone dicho requisito.
  - *Alumno:* El alumno decide que el requisito debería añadirse.
- **Verificabilidad:** Debe ser posible comprobar incuestionablemente que el requisito de usuario se ha incorporado al diseño. Los posibles valores que puede tomar son:
  - *Alta:* Se puede comprobar de manera inmediata.
  - *Media:* Es posible verificar que el requisito está añadido.
  - *Baja:* No es sencillo determinar si el requisito está presente en el diseño.
- **Estabilidad:** Indica si el requisito está abierto a modificaciones a lo largo de la vida esperada del software. Los posibles valores que recibirá este campo son:
  - *No modificable:* El requisito no se modificará durante la vida del sistema.
  - *Modificable:* El requisito está abierto a modificaciones puntuales.

La tabla 1 muestra la plantilla genérica que recogerá los requisitos de usuario y que contiene los atributos definidos anteriormente.

IDENTIFICADOR: RU-xx			
Nombre:			
Descripción:			
Prioridad:	☐ Alta	☐ Media	☐ Baja
Necesidad:	☐ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☐No modificable	
Fuente:	☐ Tutor	☐ Alumno	

Tabla 1: Plantilla de Requisitos de Usuario

4.3.1 REQUISITOS DE CAPACIDAD

Los requisitos de capacidad describen las funciones y operaciones que necesitan los usuarios para resolver un problema o lograr un objetivo. A continuación se recogen los requisitos de capacidad del sistema:

IDENTIFICADOR: RU-01			
Nombre:	Modo ocultación de mensajes.		
Descripción:	Un usuario puede acceder al modo de ocultación de mensajes en imágenes desde la interfaz principal del programa.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑ No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 2: Requisito de usuario 1.

IDENTIFICADOR: RU-02			
Nombre:	Modo extracción de mensajes.		
Descripción:	Un usuario puede acceder al modo de extracción de mensajes ocultados en imágenes desde la interfaz principal del programa.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑ No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 3: Requisito de usuario 2.

IDENTIFICADOR: RU-03			
Nombre:	Escoger imagen para ocultar.		
Descripción:	El usuario puede seleccionar una imagen de la galería desde el modo de ocultación.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑ No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 4: Requisito de usuario 3.

IDENTIFICADOR: RU-04			
Nombre:	Escribir mensaje para ocultar.		
Descripción:	El usuario es capaz de escribir un mensaje para ocultarlo.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑ No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 5: Requisito de usuario 4.

IDENTIFICADOR: RU-05			
Nombre:	Contador de caracteres.		
Descripción:	Mientras el usuario está escribiendo un mensaje oculto se le mostrará en todo momento el número de caracteres que lleva escritos.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 6: Requisito de usuario 5.

IDENTIFICADOR: RU-06			
Nombre:	Escribir contraseña para cifrar.		
Descripción:	El usuario tiene que escribir una contraseña para cifrar el mensaje escrito.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑ No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 7: Requisito de usuario 6.

IDENTIFICADOR: RU-07			
Nombre:	Escoger color en la imagen.		
Descripción:	El usuario podrá seleccionar si la foto resultante la desea en color o en blanco y negro.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 8: Requisito de usuario 7.

IDENTIFICADOR: RU-08			
Nombre:	Escoger modo de envío.		
Descripción:	Se podrá escoger el modo de envío de la imagen portadora resultante.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 9: Requisito de usuario 8.

IDENTIFICADOR: RU-09			
Nombre:	Botón ayuda.		
Descripción:	A la hora de seleccionar el modo de envío de la imagen aparecerá un botón de ayuda para describir las opciones posibles.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 10: Requisito de usuario 9.

IDENTIFICADOR: RU-10			
Nombre:	Enviar imagen por WhatsApp.		
Descripción:	La imagen que contiene el mensaje secreto debe poderse enviar a través de WhatsApp.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 11: Requisito de usuario 10.

IDENTIFICADOR: RU-11			
Nombre:	Enviar imagen por Email.		
Descripción:	Otro modo de envío debe consistir en envío de la imagen por email.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 12: Requisito de usuario 11.

IDENTIFICADOR: RU-12			
Nombre:	Enviar imagen por Line.		
Descripción:	El usuario puede enviar la imagen a la aplicación Line.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 13: Requisito de usuario 12.

IDENTIFICADOR: RU-13			
Nombre:	Enviar imagen por Spotbros.		
Descripción:	Es posible enviar la imagen a la aplicación Spotbros.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 14: Requisito de usuario 13.

IDENTIFICADOR: RU-14			
Nombre:	Enviar la imagen por MMS.		
Descripción:	Se puede enviar la imagen resultante por mensaje multimedia.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 15: Requisito de usuario 14.

IDENTIFICADOR: RU-15			
Nombre:	Progreso ocultación.		
Descripción:	Se mostrará una barra de progreso mientras se oculta el mensaje en la imagen.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 16: Requisito de usuario 15.

IDENTIFICADOR: RU-16			
Nombre:	Imágenes generadas.		
Descripción:	Las imágenes generadas por el programa se guardarán en la carpeta “Estegano” dentro de la galería.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 17: Requisito de usuario 16.

IDENTIFICADOR: RU-17			
Nombre:	Selecciona imagen para extraer el mensaje.		
Descripción:	El usuario, desde el modo de descifrado, podrá seleccionar una imagen de la galería.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑ No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 18: Requisito de usuario 17.

IDENTIFICADOR: RU-18			
Nombre:	Introducir contraseña para descifrar.		
Descripción:	El usuario podrá introducir la contraseña con la que se cifró el mensaje para poder extraerlo desde el modo de descifrado.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑ No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 19: Requisito de usuario 18.

IDENTIFICADOR: RU-19			
Nombre:	Mostrar mensaje descifrado.		
Descripción:	El mensaje descifrado debe mostrarse por pantalla.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 20: Requisito de usuario 19.

IDENTIFICADOR: RU-20			
Nombre:	Comprobar actualizaciones.		
Descripción:	Debe poderse comprobar si existen nuevas actualizaciones de la aplicación desde el menú de la interfaz principal.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 21: Requisito de usuario 20.

IDENTIFICADOR: RU-21			
Nombre:	Navegación sencilla.		
Descripción:	La interfaz de los modos de ocultación y extracción será amigable e intuitiva. Se deberán seguir los pasos que se propongan como si fuera un asistente. Y todas las opciones seleccionadas quedarán visibles, excepto la contraseña.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 22: Requisito de usuario 21.

IDENTIFICADOR: RU-22			
Nombre:	Menú compartir.		
Descripción:	La aplicación debe aparecer como una opción en el menú “compartir” y “enviar” para imágenes en Android.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 23: Requisito de usuario 22.

4.3.2 REQUISITOS DE RESTRICCIÓN

Los requisitos de restricción son aquellos que recogen algún tipo de limitación en la forma de realizar las funciones descritas por los propios usuarios. Asimismo, especifican cómo resolver el problema o cómo se debe alcanzar el objetivo. A continuación se recogen los requisitos de restricción del sistema:

IDENTIFICADOR: RU-23			
Nombre:	Longitud mensaje oculto.		
Descripción:	La longitud máxima del mensaje oculto ha de ser de 1000 caracteres.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 24: Requisito de usuario 23.

IDENTIFICADOR: RU-24			
Nombre:	Longitud contraseña.		
Descripción:	La contraseña tiene que ser como mínimo de un carácter, por cuestiones de usabilidad, y sin máximo especificado.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 25: Requisito de usuario 24.

IDENTIFICADOR: RU-25			
Nombre:	No mostrar contraseña.		
Descripción:	La contraseña deberá ser ocultada con asteriscos tras introducirla.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 26: Requisito de usuario 25.

IDENTIFICADOR: RU-26			
Nombre:	Contraseña para descifrado.		
Descripción:	Debe introducirse la misma contraseña con la que se ocultó el mensaje en la imagen para poder recuperarlo.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 27: Requisito de usuario 26.

IDENTIFICADOR: RU-27			
Nombre:	Seguridad.		
Descripción:	Los algoritmos de cifrado y ocultación deben ser seguros y no considerarse como “rotos” o vulnerables.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☑ Tutor	☐ Alumno	

Tabla 28: Requisito de usuario 27.

IDENTIFICADOR: RU-28			
Nombre:	Aplicaciones instaladas.		
Descripción:	El programa no mostrará los modos de envío que no pueden ser utilizados por el usuario si no tiene instalada la aplicación correspondiente.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 29: Requisito de usuario 28.

IDENTIFICADOR: RU-29			
Nombre:	Tiempo para ocultar mensaje.		
Descripción:	La aplicación debe tardar menos de 1 minuto en ocultar el mensaje en la imagen.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☐ Media	☑ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 30: Requisito de usuario 29.

IDENTIFICADOR: RU-30			
Nombre:	Tiempo para descifrar mensaje.		
Descripción:	La aplicación debe tardar menos de 20 segundos en extraer el mensaje oculto de la imagen.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☐ Media	☑ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 31: Requisito de usuario 30.

IDENTIFICADOR: RU-31			
Nombre:	Conectividad a Internet.		
Descripción:	El programa no debe necesitar conexión a Internet para ocultar mensajes y extraerlos, únicamente para comprobar actualizaciones periódicamente.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 32: Requisito de usuario 31.

IDENTIFICADOR: RU-32			
Nombre:	Idiomas.		
Descripción:	La aplicación debe estar traducida en castellano e inglés.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	☐ Tutor	☑ Alumno	

Tabla 33: Requisito de usuario 32.

4.4 CASOS DE USO

En esta tarea se van a detallar los casos de uso del sistema, que se obtienen de los requisitos de usuario definidos en el apartado anterior. Cada caso de uso satisface uno o varios requisitos de usuario, concretamente aquellos que estén directamente involucrados con la actividad que resuelve el caso de uso. Los casos de uso describen las posibles acciones que el usuario final puede realizar sobre el sistema.

En primer lugar se ilustrarán los casos de uso principales en un diagrama general sencillo. Posteriormente se construirán dos diagramas adicionales más detallados que contienen todos los casos de uso descritos en los requisitos, de esta manera se sigue un proceso de análisis incremental que se completará con la descripción detallada de cada caso de uso presente en los diagramas.

Para completar los casos de uso se debe especificar en una tabla para cada uno:

- **Identificador:** Cada caso de uso incluirá un identificador unívoco que facilitará su posterior trazabilidad. Estará compuesto de dos siglas, que se refieren a los casos de uso (CU), y dos números, que se emplearán para numerarlos.
- **Caso de uso:** Este campo resumirá la funcionalidad del caso de uso.
- **Actores:** Se indicará qué actor interactúa con este caso de uso.
- **Objetivo:** Acción principal que realiza el caso de uso.
- **Descripción:** Breve descripción de cómo un actor interactúa con el sistema y cuál es la respuesta que el sistema le ofrece.
- **Precondiciones y postcondiciones:** Es necesario definir las condiciones que se deben cumplir para poder realizar una acción, y en qué estado queda el sistema tras realizar la misma.
- **Flujo normal:** Establece el orden de acciones necesarias para llegar al caso de uso indicado.
- **Flujo alternativo:** Si lo hay, éste describe caminos alternativos para alcanzar el mismo caso de uso.
- **Referencias:** Requisitos de usuario con los que esté relacionado.

La tabla 34 muestra la plantilla genérica que recogerá los casos de uso y que contiene los atributos definidos anteriormente.

IDENTIFICADOR: CU-xx			
Caso de uso:		**Actores:**	
Objetivo:			
Descripción			
Precondiciones:			
Postcondiciones:			
Flujo normal:			
Flujo alternativo:			
Referencias:			

Tabla 34: Plantilla de casos de uso.

4.4.1 CASO DE USO GENERAL

A continuación se va a mostrar el diagrama general de casos de uso para la aplicación:

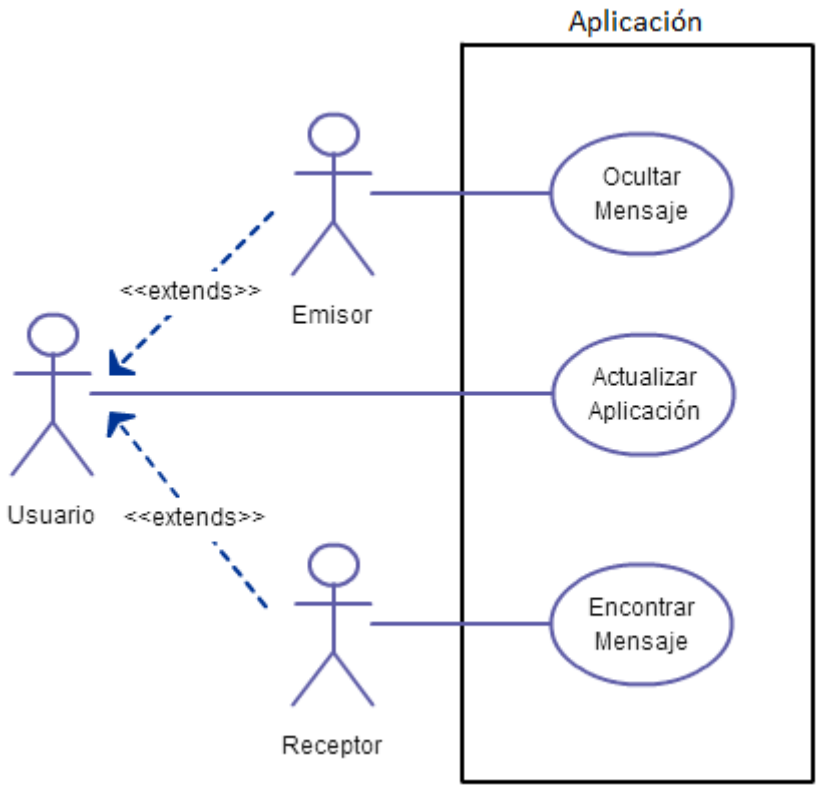


Ilustración 8: Diagrama de caso de uso principal

Como se muestra en la figura 8, existen 3 casos de uso generales: ocultar un mensaje en una imagen, encontrar el mensaje, y actualizar la aplicación.

La figura anterior muestra a 3 actores, aunque 2 de ellos derivan del usuario principal, a continuación se explican sus roles:

- Usuario: Es aquel que va a utilizar la aplicación, como la aplicación tiene 2 módulos principales bien diferenciados ha sido necesario derivar de éste usuario otros dos actores que dependiendo del módulo escogido tomarán un rol u otro. Sin embargo el usuario desde la interfaz principal puede comprobar si existen nuevas actualizaciones.
- Emisor: Se trata de una especificación del usuario principal. El caso de uso principal que maneja este "usuario emisor" es el de ocultar un mensaje oculto en una imagen.
- Receptor: Este actor también deriva del usuario principal, su caso de uso corresponde a la extracción del mensaje ocultado en una imagen.

Los casos de uso "ocultar mensaje" y "encontrar mensaje" se van a desglosar en profundidad en las siguientes secciones, ahora sólo se contemplará el acceso a dichos modos:

IDENTIFICADOR: CU-01			
Caso de uso:	Ocultar mensaje.	**Actores:**	Emisor.
Objetivo:	Acceder al modo de ocultación.		
Descripción	El usuario selecciona el modo de ocultación desde la interfaz principal.		
Precondiciones:	Abrir aplicación.		
Postcondiciones:	Acceso al modo de ocultación.		
Flujo normal:	1. Abrir la aplicación. 2. Acceder al modo de ocultación.		
Flujo alternativo:	(1). El usuario accede al modo de ocultación desde el menú "Compartir con" o "Enviar a" de las imágenes en Android.		
Referencias:	RU-01, RU-22		

Tabla 35: Caso de uso 1.

IDENTIFICADOR: CU-02			
Caso de uso:	Encontrar mensaje.	**Actores:**	Receptor.
Objetivo:	Acceder al modo de extracción.		
Descripción	El usuario selecciona el modo de extracción desde la interfaz principal.		
Precondiciones:	Abrir aplicación.		
Postcondiciones:	Acceder al modo de extracción.		
Flujo normal:	1. Abrir la aplicación. 2. Acceder al modo de recuperación del mensaje.		
Flujo alternativo:	(1). El usuario accede al modo de extracción desde el menú "Compartir con" o "Enviar a" de las imágenes en Android.		
Referencias:	RU-02, RU-22		

Tabla 36: Caso de uso 2.

IDENTIFICADOR: CU-03			
Caso de uso:	Actualizar aplicación.	**Actores:**	Usuario.
Objetivo:	Comprobar si hay nuevas actualizaciones.		
Descripción	El usuario selecciona en el menú de la aplicación la opción de actualizar.		
Precondiciones:	Abrir la aplicación.		
Postcondiciones:	Respuesta a la petición de actualizar.		
Flujo normal:	1. Abrir la aplicación. 2. Acceder al menú. 3. Seleccionar opción de actualizar.		
Flujo alternativo:	(2). Periódicamente se comprueba de forma automática si hay nuevas actualizaciones, y si las hay se muestra al usuario por pantalla.		
Referencias:	RU-20, RU-31		

Tabla 37: Caso de uso 3.

4.4.2 CASOS DE USO MÓDULO OCULTACIÓN

A continuación, en la figura 9, se muestra un diagrama más completo, en el que se desglosan los casos de uso para el módulo de ocultación de mensajes.

En esta ocasión el único actor presente en el escenario es el emisor, pues es él el encargado de ocultar el mensaje y enviarlo. Se puede observar que el proceso de ocultación está formado por distintos casos de uso, cada uno de ellos representa una acción que se debe completar para alcanzar el objetivo.

Se han incluido las posibles elecciones dentro de los casos de uso de selección del color y modo de envío, en el caso del primero se puede escoger que la imagen resultante esté en color o en blanco y negro; y en el segundo se desglosan los posibles métodos para enviar la imagen.

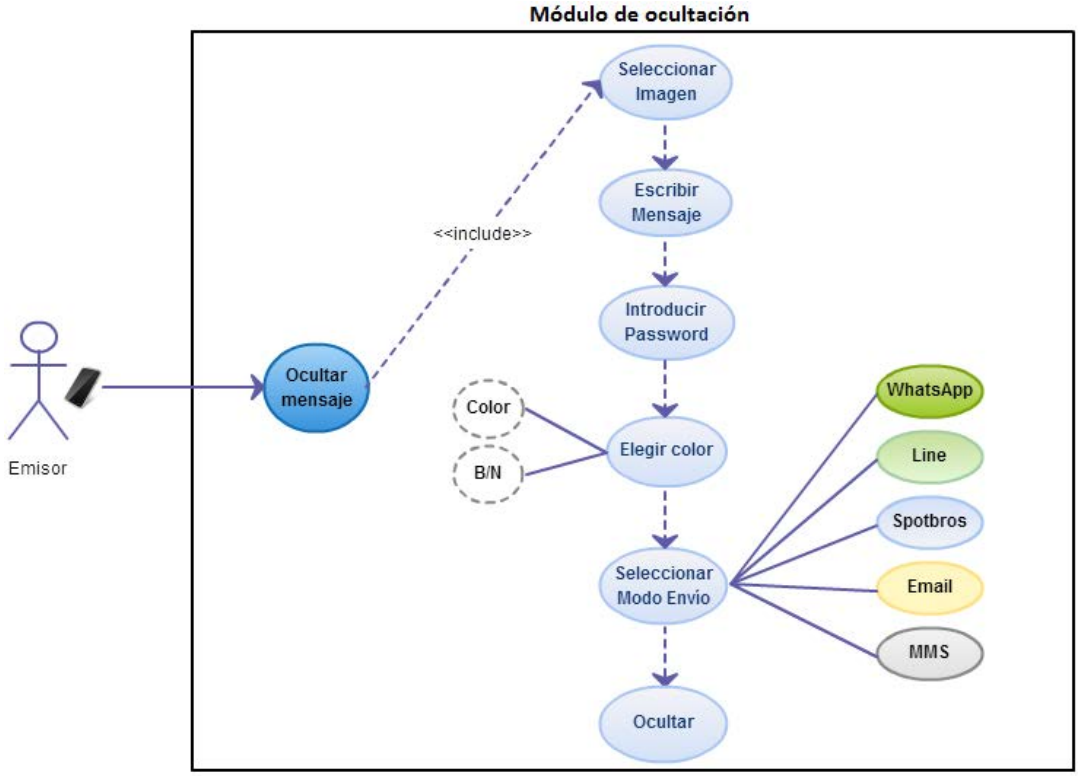


Ilustración 9: Diagrama de casos de uso para el módulo de ocultación

En las siguientes tablas se explica de manera completa cada uno de los casos de uso mostrados en la ilustración anterior:

IDENTIFICADOR: CU-04			
Caso de uso:	Seleccionar imagen.	**Actores:**	Emisor.
Objetivo:	Escoger una imagen de la galería.		
Descripción	El usuario elige la imagen en la que desea introducir el mensaje secreto.		
Precondiciones:	Acceder al modo de ocultación.		
Postcondiciones:	Imagen seleccionada.		
Flujo normal:	1. Acceder al modo de ocultación desde la interfaz principal. 2. Escoger una imagen de la galería.		
Flujo alternativo:	(1). En el menú “compartir” de una imagen abrir “Ocultar secreto”. (3). Si se selecciona una imagen con formato no soportado el sistema avisa al usuario y no continua.		
Referencias:	RU-03, RU-21		

Tabla 38: Caso de uso 4.

IDENTIFICADOR: CU-05			
Caso de uso:	Escribir mensaje.	**Actores:**	Emisor.
Objetivo:	Escribir el mensaje secreto.		
Descripción	El usuario compone el mensaje que desea ocultar.		
Precondiciones:	Seleccionar imagen.		
Postcondiciones:	Mensaje escrito.		
Flujo normal:	1. Escoger una imagen de la galería. 2. Escribir el mensaje secreto.		
Flujo alternativo:	(2). Si el usuario supera los 1000 caracteres de texto la aplicación no le dejará escribir más.		
Referencias:	RU-04, RU-05, RU-21, RU-23		

Tabla 39 Caso de uso 5.

IDENTIFICADOR: CU-06			
Caso de uso:	Introducir password.	**Actores:**	Emisor.
Objetivo:	Escribir una contraseña.		
Descripción	El usuario inserta la contraseña con la que desea ocultar el mensaje y cifrarlo.		
Precondiciones:	Escribir mensaje a ocultar.		
Postcondiciones:	Contraseña introducida.		
Flujo normal:	1. Escoger una imagen de la galería. 2. Escribir el mensaje secreto. 3. Introducir una contraseña.		
Flujo alternativo:			
Referencias:	RU-06, RU-21, RU-24, RU-25		

Tabla 40: Caso de uso 6.

IDENTIFICADOR: CU-07			
Caso de uso:	Elegir color.	**Actores:**	Emisor.
Objetivo:	Escoger si la imagen resultante estará en color o en blanco y negro.		
Descripción	El usuario selecciona el modo de color que prefiera.		
Precondiciones:	Introducir una contraseña.		
Postcondiciones:	Modo de color seleccionado.		
Flujo normal:	1. Seleccionar imagen de galería. 2. Escribir mensaje secreto. 3. Introducir una contraseña. 4. Escoger color de imagen.		
Flujo alternativo:			
Referencias:	RU-07, RU-21		

Tabla 41: Caso de uso 7.

IDENTIFICADOR: CU-08			
Caso de uso:	Seleccionar modo envío.	**Actores:**	Emisor.
Objetivo:	Escoger la aplicación a la que se quiere enviar la imagen resultante.		
Descripción	El emisor escoge de una lista de aplicaciones la que quiere que reciba la imagen.		
Precondiciones:	Seleccionar modo de color.		
Postcondiciones:	Modo de envío fijado.		
Flujo normal:	1. Seleccionar imagen de galería. 2. Escribir mensaje secreto. 3. Introducir una contraseña. 4. Escoger color de imagen. 5. Elegir modo de envío.		
Flujo alternativo:			
Referencias:	RU-08, RU-09, RU-10, RU-11, RU-12, RU-13, RU-14, RU-21, RU-28		

Tabla 42: Caso de uso 8.

IDENTIFICADOR: CU-09			
Caso de uso:	Ocultar.	**Actores:**	Emisor.
Objetivo:	La aplicación oculta el mensaje en la imagen seleccionada según las opciones introducidas en los pasos anteriores.		
Descripción	El usuario ha completado el asistente rellenando todas las opciones y pulsa un botón para proceder con la ocultación.		
Precondiciones:	Acceder al modo de ocultación.		
Postcondiciones:	Mensaje secreto oculto en una imagen.		
Flujo normal:	1. Seleccionar imagen de galería. 2. Escribir mensaje secreto. 3. Introducir una contraseña. 4. Escoger color de imagen. 5. Elegir modo de envío. 6. Pulsar el botón “Ocultar”.		
Flujo alternativo:	(6). Si la ocultación falla se notifica al usuario el error.		
Referencias:	RU-15, RU-16, RU-21, RU-27 RU-29		

Tabla 43: Caso de uso 9.

4.4.3 CASOS DE USO MÓDULO EXTRACCIÓN DEL MENSAJE

En la figura 10 se representa el otro escenario posible, en el que el actor "receptor" pretende recuperar el mensaje oculto en una imagen. Se trata de un diagrama mucho más sencillo que el anterior, en el que el caso de uso general de "encontrar mensaje" se desglosa en 3 casos de uso: seleccionar imagen, introducir contraseña, y mostrar mensaje finalmente.

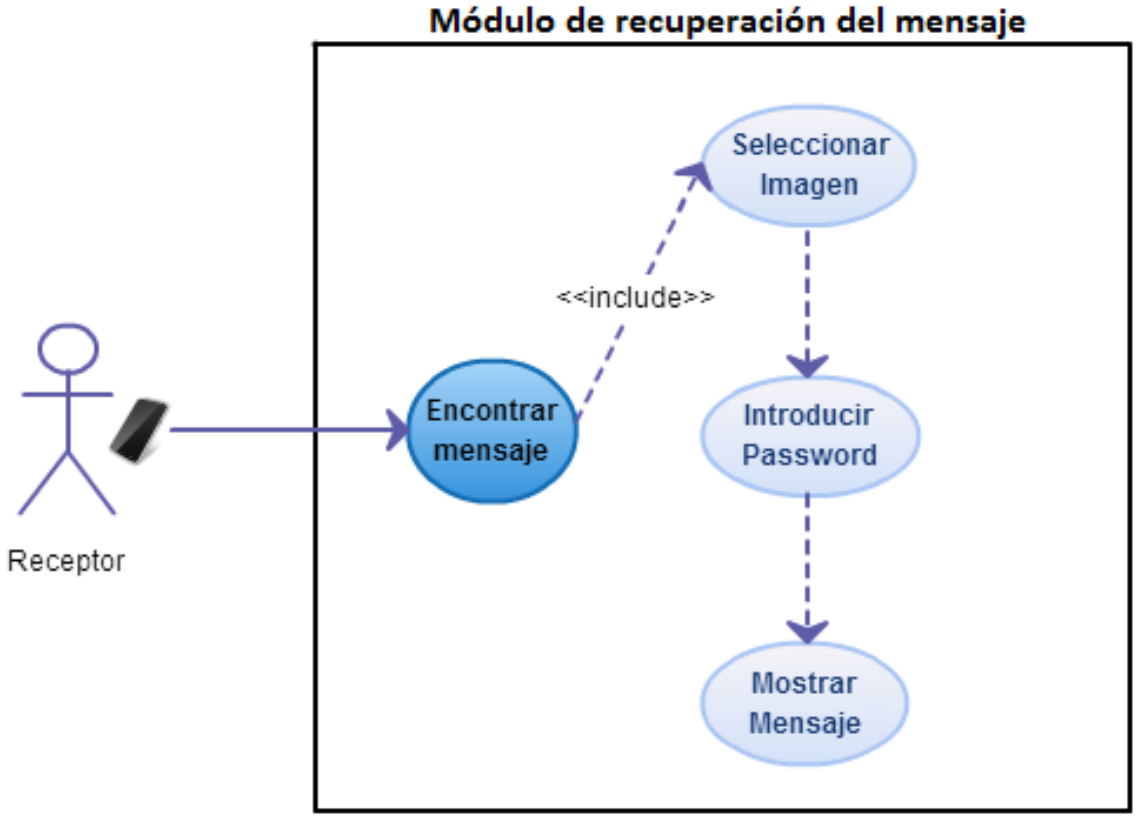


Ilustración 10: Diagrama de casos de uso para el módulo de recuperación

Los casos de usos detallados para el diagrama anterior se listan en las siguientes tablas:

IDENTIFICADOR: CU-10			
Caso de uso:	Seleccionar imagen.	**Actores:**	Receptor.
Objetivo:	El receptor selecciona la imagen que tiene información oculta.		
Descripción	Se navega a través de las imágenes de la galería para seleccionar aquella portadora de un mensaje secreto que se quiere recuperar.		
Precondiciones:	Acceder al modo "encontrar".		
Postcondiciones:	Imagen seleccionada.		
Flujo normal:	3. Acceder al modo de encontrar mensaje. 4. Seleccionar imagen de galería.		
Flujo alternativo:	(1). En el menú "compartir" de una imagen abrir "Encontrar secreto". (2). Si se selecciona una imagen con formato no soportado el sistema avisa al usuario y no continua.		
Referencias:	RU-17, RU-21		

Tabla 44: Caso de uso 10.

IDENTIFICADOR: CU-11			
Caso de uso:	Introducir password.	**Actores:**	Receptor.
Objetivo:	El receptor introduce la contraseña correcta.		
Descripción	Se escribe la contraseña con la que fue ocultada y cifrada el mensaje secreto en la imagen.		
Precondiciones:	Seleccionar imagen.		
Postcondiciones:	Contraseña introducida.		
Flujo normal:	1. Acceder al modo de encontrar mensaje. 2. Seleccionar imagen de galería. 3. Introducir la contraseña.		
Flujo alternativo:			
Referencias:	RU-18, RU-21, RU-24, RU-25, RU-26		

Tabla 45: Caso de uso 11.

IDENTIFICADOR: CU-12			
Caso de uso:	Mostrar mensaje.	**Actores:**	Receptor.
Objetivo:	El receptor visualiza el mensaje secreto.		
Descripción	Se pulsa el botón “encontrar” para que el programa, utilizando la imagen seleccionada y la contraseña introducida, extraiga el mensaje oculto.		
Precondiciones:	Introducir contraseña.		
Postcondiciones:	Mensaje secreto mostrado.		
Flujo normal:	1. Acceder al modo de encontrar mensaje. 2. Seleccionar imagen de galería. 3. Introducir la contraseña. 4. Pulsar botón “encontrar”.		
Flujo alternativo:	(4). Si la contraseña introducía es inválida, o la imagen no porta ningún mensaje oculto se mostrará al usuario un mensaje de error.		
Referencias:	RU-19, RU-21, RU-30		

Tabla 46: Caso de uso 12.

4.5 REQUISITOS DE SOFTWARE

Los requisitos de software definen qué tiene que hacer el producto, en otras palabras, describen la funcionalidad que el sistema tendrá. Se obtienen del modelo de casos de uso y de los requisitos de usuario.

La especificación de requisitos de software se divide en dos grandes grupos, los requisitos de software funcionales y los no funcionales.

Los requisitos de software funcionales especifican qué tiene que hacer el software, su propósito, y se obtienen de los casos de uso (que a su vez están derivados de los requisitos de usuario).

Los requisitos de software no funcionales especifican cómo deben llevarse a cabo las funcionalidades del sistema. Éstos se pueden subdividir a su vez en las siguientes categorías:

- Requisitos de operación: Indican cómo va a realizar el sistema las tareas para las que ha sido construido.
- Requisitos de interfaz: Especifican el modo en que interactúan y se comunican las interfaces y diferentes módulos del sistema.
- Requisitos de rendimiento: Especifican los valores numéricos que representan el rendimiento esperado del sistema.
- Requisitos de recursos: Especifican las limitaciones y recursos mínimos que el sistema requiere para su correcto funcionamiento.
- Requisitos de comprobación: Especifican las limitaciones que afectan a cómo el software debe verificar los datos de entrada y salida.
- Requisitos de seguridad: Especifican los requisitos para asegurar el sistema contra amenazas en la confidencialidad, integridad y disponibilidad.
- Requisitos de mantenimiento: Indican la facilidad que tendrá el sistema para reparar los defectos o incorporar nuevas funcionalidades.

Para redactar los requisitos de software se empleará la plantilla de la tabla 47, en la que se especifican los siguientes campos:

- **Identificación:** Cada requisito de usuario incluirá un identificador unívoco que facilitará su posterior trazabilidad. Estará compuesto de dos siglas, que se refieren a los requisitos de software (RS), y dos números, que se emplearán para numerarlos.
- **Descripción:** Breve descripción del requisito software.
- **Prioridad:** Refleja la prioridad establecida para que el desarrollador puede decidir la planificación de la producción. Los posibles valores que puede recibir este atributo son:
    - *Alta:* Se trata de un requisito que debe ser añadido antes que los de menor prioridad.
    - *Media:* Tras completar los requisitos de prioridad alta se implementan éstos antes que los que poseen prioridad baja.
    - *Baja:* Son los últimos en añadirse.

- **Necesidad:** Definen la importancia del requisito. Los valores que puede tomar este campo son:
  - *Esencial:* El requisito no será negociable y debe ser implementado.
  - *Deseable:* Es importante implementar dicho requisito pero no obligatorio.
  - *Opcional:* Se podrá implementar si las circunstancias lo permiten.
- **Fuente:** Referencia al requisito o requisitos de usuario de los que parten los requisitos software
- **Verificabilidad:** Debe ser posible comprobar incuestionablemente que el requisito se ha incorporado al diseño. Los posibles valores que puede tomar son:
  - *Alta:* Se puede comprobar de manera inmediata.
  - *Media:* Es posible verificar que el requisito está añadido.
  - *Baja:* No es sencillo determinar si el requisito está presente en el sistema.
- **Estabilidad:** Indica si el requisito está abierto a modificaciones a lo largo de la vida esperada del software. Los posibles valores que recibirá este campo son:
  - *No modificable:* El requisito no se modificará durante la vida del sistema.
  - *Modificable:* El requisito está abierto a modificaciones durante el diseño o implementación del sistema.

La tabla 47 muestra la plantilla genérica que recogerá los requisitos de software y que contiene los atributos definidos anteriormente.

IDENTIFICADOR: RS-xx			
Descripción:			
Prioridad:	☐ Alta	☐ Media	☐ Baja
Necesidad:	☐ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☐No modificable	
Fuente:			

Tabla 47: Plantilla de requisitos de software.

4.5.1 REQUISITOS FUNCIONALES

A continuación se listan los requisitos que detallan qué hace el sistema:

IDENTIFICADOR: RS-01			
Descripción:	El sistema deberá permitir al usuario acceder al modo de ocultación de mensajes en imágenes.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-01		

Tabla 48: Requisito de software 1.

IDENTIFICADOR: RS-02			
Descripción:	El sistema deberá permitir al usuario acceder al modo de extracción de mensajes en imágenes.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-02		

Tabla 49: Requisito de software 2.

IDENTIFICADOR: RS-03			
Descripción:	El sistema tiene que permitir al usuario seleccionar una imagen de la galería desde el modo de ocultación de mensajes.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-03		

Tabla 50: Requisito de software 3.

IDENTIFICADOR: RS-04			
Descripción:	El sistema tiene que permitir al usuario escribir un mensaje de texto.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-04		

Tabla 51: Requisito de software 4.

IDENTIFICADOR: RS-05			
Descripción:	El sistema tiene que permitir al usuario escribir hasta 1000 caracteres como máximo.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-23		

Tabla 52: Requisito de software 5.

IDENTIFICADOR: RS-06			
Descripción:	El sistema deberá mostrar un contador del número de caracteres que se han escrito en el mensaje.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-05		

Tabla 53: Requisito de software 6.

IDENTIFICADOR: RS-07			
Descripción:	El sistema deberá permitir que el usuario introduzca una contraseña.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-06		

Tabla 54: Requisito de software 7.

IDENTIFICADOR: RS-08			
Descripción:	El sistema tiene que permitir al usuario seleccionar si la imagen resultante del proceso de ocultación ha de estar en blanco y negro, o en color.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-07		

Tabla 55: Requisito de software 8.

IDENTIFICADOR: RS-09			
Descripción:	El sistema tiene que permitir al usuario escoger el modo de envío de la imagen resultante del proceso de ocultación.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-08		

Tabla 56: Requisito de software 9.

IDENTIFICADOR: RS-10			
Descripción:	El sistema permitirá escoger el modo de envío mediante WhatsApp siempre que esté instalado.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-10		

Tabla 57: Requisito de software 10.

IDENTIFICADOR: RS-11			
Descripción:	El sistema permitirá escoger el modo de envío mediante Line siempre que esté instalado.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-12		

Tabla 58: Requisito de software 11.

IDENTIFICADOR: RS-12			
Descripción:	El sistema permitirá escoger el modo de envío mediante Spotbros siempre que esté instalado.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-13		

Tabla 59: Requisito de software 12.

IDENTIFICADOR: RS-13			
Descripción:	El sistema permitirá escoger el modo de envío mediante email.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-11		

Tabla 60: Requisito de software 13.

IDENTIFICADOR: RS-14			
Descripción:	El sistema permitirá escoger el modo de envío mediante MMS.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-14		

Tabla 61: Requisito de software 14.

IDENTIFICADOR: RS-15			
Descripción:	El sistema tiene que permitir al usuario ocultar un mensaje en una imagen usando los datos que ha introducido.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-04 RU-06, RU-07, RU-08		

Tabla 62: Requisito de software 15.

IDENTIFICADOR: RS-16			
Descripción:	El sistema tiene que mostrar una barra de progreso durante el proceso de ocultación de mensajes.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-15		

Tabla 63: Requisito de software 16.

IDENTIFICADOR: RS-17			
Descripción:	El sistema tiene que guardar las imágenes generadas con el programa en la carpeta “Estegano” dentro de la galería.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-16		

Tabla 64: Requisito de software 17.

IDENTIFICADOR: RS-18			
Descripción:	El sistema tiene que permitir al usuario escoger una imagen de la galería desde el modo “encontrar”.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-17		

Tabla 65: Requisito de software 18.

IDENTIFICADOR: RS-19			
Descripción:	El sistema tiene que permitir al usuario escribir una contraseña para extraer el mensaje oculto en una imagen.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-18		

Tabla 66: Requisito de software 19.

IDENTIFICADOR: RS-20			
Descripción:	El sistema tiene que permitir al usuario extraer un mensaje oculto a partir de las opciones seleccionadas.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-17, RU-18		

Tabla 67: Requisito de software 20.

IDENTIFICADOR: RS-21			
Descripción:	El sistema tiene que mostrar el mensaje oculto por pantalla una vez ha sido extraído.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-19		

Tabla 68: Requisito de software 21.

IDENTIFICADOR: RS-22			
Descripción:	El sistema tiene que permitir al usuario comprobar si existen nuevas actualizaciones de la aplicación.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-20		

Tabla 69: Requisito de software 22.

4.5.2 REQUISITOS DE OPERACIÓN

IDENTIFICADOR: RS-23			
Descripción:	El sistema deberá permitir al usuario acceder desde la pantalla principal de la interfaz al modo de ocultación de mensajes.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-01		

Tabla 70: Requisito de software 23.

IDENTIFICADOR: RS-24			
Descripción:	El sistema deberá permitir al usuario acceder desde la pantalla principal de la interfaz al modo de extracción de mensajes.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-02		

Tabla 71: Requisito de software 24.

IDENTIFICADOR: RS-25			
Descripción:	Se mostrará un menú con la opción de actualizar aplicación cuando se presione el botón “menú” de Android en la pantalla principal de la aplicación.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-02		

Tabla 72: Requisito de software 25.

IDENTIFICADOR: RS-26			
Descripción:	El sistema debe permitir al usuario navegar por la interfaces de los módulos de ocultación y extracción de mensajes para corregir o modificar los campos mostrados, para ello se hará uso del botón “atrás” que dispone Android y de botones para avanzar.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-21		

Tabla 73: Requisito de software 26.

IDENTIFICADOR: RS-27			
Descripción:	El sistema mostrará botón para elegir una imagen de la galería en los modos de ocultación y recuperación del mensaje.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-15, RU-19		

Tabla 74: Requisito de software 27.

IDENTIFICADOR: RS-28			
Descripción:	El sistema mostrará dos opciones para escoger el modelo de color de la imagen: - Color. - Blanco y negro.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-07		

Tabla 75: Requisito de software 28.

IDENTIFICADOR: RS-29			
Descripción:	El sistema mostrará un desplegable para la opción modo de envío con las siguientes opciones (dependiendo de si están disponibles o no): - WhatsApp. - Line. - Spotbros. - MMS. - Email.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-08, RU-10, RU-11, RU-12, RU-13, RU-14		

Tabla 76: Requisito de software 29.

IDENTIFICADOR: RS-30			
Descripción:	El sistema mostrará un botón de ayuda a la hora de seleccionar el modo de envío. Éste suministrará ayuda por pantalla al usuario en caso de pulsarlo.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-09		

Tabla 77: Requisito de software 30.

IDENTIFICADOR: RS-31			
Descripción:	El sistema debe enviar la imagen al programa que se halla seleccionado en el modo de envío.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-08, RU-15		

Tabla 78: Requisito de software 31.

IDENTIFICADOR: RS-32			
Descripción:	El sistema mostrará un botón “ocultar” en la parte inferior del modo de ocultación cuando el usuario haya rellenado todos los campos requeridos. Éste botón realizará la acción de introducir el mensaje secreto en la imagen.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-15		

Tabla 79: Requisito de software 32.

IDENTIFICADOR: RS-33			
Descripción:	El sistema mostrará un botón “encontrar” en la parte inferior del modo de recuperación del mensaje cuando el usuario haya rellenado todos los campos requeridos. Éste botón será el encargado de llamar a los métodos de extracción del mensaje.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-19		

Tabla 80: Requisito de software 33.

4.5.3 REQUISITOS DE INTERFAZ

IDENTIFICADOR: RS-34			
Descripción:	El sistema tiene que permitir al usuario poder seleccionar imágenes con formato JPEG y PNG.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-03, RU-17		

Tabla 81: Requisito de software 34.

IDENTIFICADOR: RS-35			
Descripción:	La aplicación podrá ser llamada de manera externa cuando se seleccione desde el menú contextual de Android “enviar a” o “compartir con” sobre imágenes.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-22		

Tabla 82: Requisito de software 35.

4.5.4 REQUISITOS DE RENDIMIENTO

IDENTIFICADOR: RS-36			
Descripción:	El sistema tiene que permitir al usuario poder seleccionar imágenes de cualquier tamaño, peso y resolución.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-03, RU-17		

Tabla 83: Requisito de software 36.

IDENTIFICADOR: RS-37			
Descripción:	El sistema debe ser capaz de ocultar un mensaje en una imagen en menos de 1 minuto, siempre que los datos introducidos hayan sido validados.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-29		

Tabla 84: Requisito de software 37.

IDENTIFICADOR: RS-38			
Descripción:	El sistema debe ser capaz de extraer un mensaje oculto en menos de 20 segundos, siempre que los datos introducidos sean válidos.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-30		

Tabla 85: Requisito de software 38.

IDENTIFICADOR: RS-39			
Descripción:	Dependiendo del idioma del dispositivo Android la aplicación se mostrará en castellano o inglés.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-31		

Tabla 86: Requisito de software 39.

4.5.5 REQUISITOS DE RECURSOS

IDENTIFICADOR: RS-40			
Descripción:	El sistema debe disponer de conexión a Internet para comprobar si existen nuevas actualizaciones disponibles.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-02, RU-22		

Tabla 87: Requisito de software 40.

IDENTIFICADOR: RS-41			
Descripción:	El dispositivo móvil debe disponer de memoria RAM suficiente para poder utilizar la aplicación sin tener excepciones de memoria insuficiente.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:			

Tabla 88: Requisito de software 41.

IDENTIFICADOR: RS-42			
Descripción:	El sistema funcionará en sistemas operativos Android a partir de la versión de firmware 2.2.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:			

Tabla 89: Requisito de software 42.

4.5.6 REQUISITOS DE SEGURIDAD

IDENTIFICADOR: RS-43			
Descripción:	El sistema tiene que emplear algoritmos de cifrado y ocultación que no se consideren "rotos" o vulnerables.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-27		

Tabla 90: Requisito de software 43.

IDENTIFICADOR: RS-44			
Descripción:	El sistema tiene que cifrar y codificar la información antes de ocultarla. Para ello utilizará la contraseña introducida.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:			

Tabla 91: Requisito de software 44.

IDENTIFICADOR: RS-45			
Descripción:	El sistema tiene que asegurar que el mensaje secreto introducido en una imagen sólo es posible descifrarlo si se conoce la contraseña empleada cuando éste se ocultó.		
Prioridad:	☐ Alta	☐ Media	☐ Baja
Necesidad:	☐ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☐No modificable	
Fuente:	RU-26		

Tabla 92: Requisito de software 44.

IDENTIFICADOR: RS-46			
Descripción:	El sistema ocultará la contraseña con asteriscos tras ser introducida por el usuario.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-25		

Tabla 93: Requisito de software 46.

IDENTIFICADOR: RS-47			
Descripción:	El sistema tiene que ser capaz de aplicar técnicas de corrección de errores para recuperar información dañada durante el envío de información.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:			

Tabla 94: Requisito de software 47.

IDENTIFICADOR: RS-48			
Descripción:	El sistema tiene que dispersar pseudo-aleatoriamente el mensaje cifrado por toda la imagen.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:			

Tabla 95: Requisito de software 48.

IDENTIFICADOR: RS-49			
Descripción:	El sistema tiene que cumplir la Ley Orgánica de Protección de Datos.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:			

Tabla 96: Requisito de software 49.

4.5.7 REQUISITOS DE VERIFICACIÓN

IDENTIFICADOR: RS-50			
Descripción:	El sistema tiene que controlar los errores que se puedan producir durante el funcionamiento de la aplicación y proporcionar los mensajes de error o advertencia adecuados.		
Prioridad:	☐ Alta	☑ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:			

Tabla 97: Requisito de software 50.

IDENTIFICADOR: RS-51			
Descripción:	El sistema tiene que comprobar que se ha seleccionado una imagen.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-03, RU-17		

Tabla 98: Requisito de software 51.

IDENTIFICADOR: RS-52			
Descripción:	El sistema tiene que comprobar si la ruta de la imagen escogida es correcta.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-03, RU-17		

Tabla 99: Requisito de software 52.

IDENTIFICADOR: RS-53			
Descripción:	El sistema tiene que comprobar que la imagen seleccionada está en un formato válido aceptado por la aplicación.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-03, RU-17		

Tabla 100: Requisito de software 53.

IDENTIFICADOR: RS-54			
Descripción:	El sistema tiene que comprobar que se ha introducido un mensaje de texto.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-04		

Tabla 101: Requisito de software 54.

IDENTIFICADOR: RS-55			
Descripción:	El sistema tiene que comprobar que el mensaje de texto introducido no contiene ningún carácter inválido.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-05		

Tabla 102: Requisito de software 55.

IDENTIFICADOR: RS-56			
Descripción:	El sistema tiene que comprobar si el mensaje introducido sobrepasa el límite de caracteres máximos permitido.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-04 RU-05		

Tabla 103: Requisito de software 56.

IDENTIFICADOR: RS-57			
Descripción:	El sistema tiene que comprobar si se ha introducido una contraseña		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-06, RU-18		

Tabla 104: Requisito de software 57.

IDENTIFICADOR: RS-58			
Descripción:	El sistema tiene que comprobar que la contraseña no contiene caracteres inválidos.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☑ Deseable	☐ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-06, RU-18		

Tabla 105: Requisito de software 58.

IDENTIFICADOR: RS-59			
Descripción:	El sistema tiene que comprobar que se ha seleccionado un modo de color.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-07		

Tabla 106: Requisito de software 59.

IDENTIFICADOR: RS-60			
Descripción:	El sistema tiene que comprobar que se ha escogido un modo de envío para la imagen resultante.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-08		

Tabla 107: Requisito de software 60.

IDENTIFICADOR: RS-61			
Descripción:	El sistema tiene que comprobar que el mensaje introducido cabe en la imagen.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:			

Tabla 108: Requisito de software 61.

IDENTIFICADOR: RS-62			
Descripción:	El sistema tiene que comprobar que la contraseña introducida para descifrar el mensaje es válida y que se corresponde con la original.		
Prioridad:	☑ Alta	☐ Media	☐ Baja
Necesidad:	☑ Esencial	☐ Deseable	☐ Opcional
Verificabilidad:	☑ Alta	☐ Media	☐ Baja
Estabilidad:	☐ Modificable	☑No modificable	
Fuente:	RU-26		

Tabla 109: Requisito de software 62.

4.5.8 REQUISITOS DE MANTENIMIENTO

IDENTIFICADOR: RS-63			
Descripción:	El sistema dispondrá de un mecanismo para comprobar periódicamente si existen nuevas actualizaciones de la aplicación, dependiendo del tipo de actualización se mostrará un mensaje por pantalla al usuario describiendo los nuevos cambios y la necesidad de realizarla.		
Prioridad:	☐ Alta	☐ Media	☑ Baja
Necesidad:	☐ Esencial	☐ Deseable	☑ Opcional
Verificabilidad:	☐ Alta	☑ Media	☐ Baja
Estabilidad:	☑ Modificable	☐No modificable	
Fuente:	RU-02		

Tabla 110: Requisito de software 63.

5 DISEÑO

El objetivo del presente capítulo es **resolver el problema** que se ha descrito y analizado anteriormente justificando cada una de las decisiones tomadas.

Para abordar el diseño de la aplicación se seguirá un **procedimiento incremental**, comenzaremos describiendo la aplicación e iremos desglosando los subsistemas que la conforman hasta llegar a los detalles más específicos de cada módulo.

5.1 ARQUITECTURA DEL SISTEMA

La arquitectura del sistema se ha dividido en 4 capas distintas en la que cada una se comunica con la que tiene inmediatamente debajo.

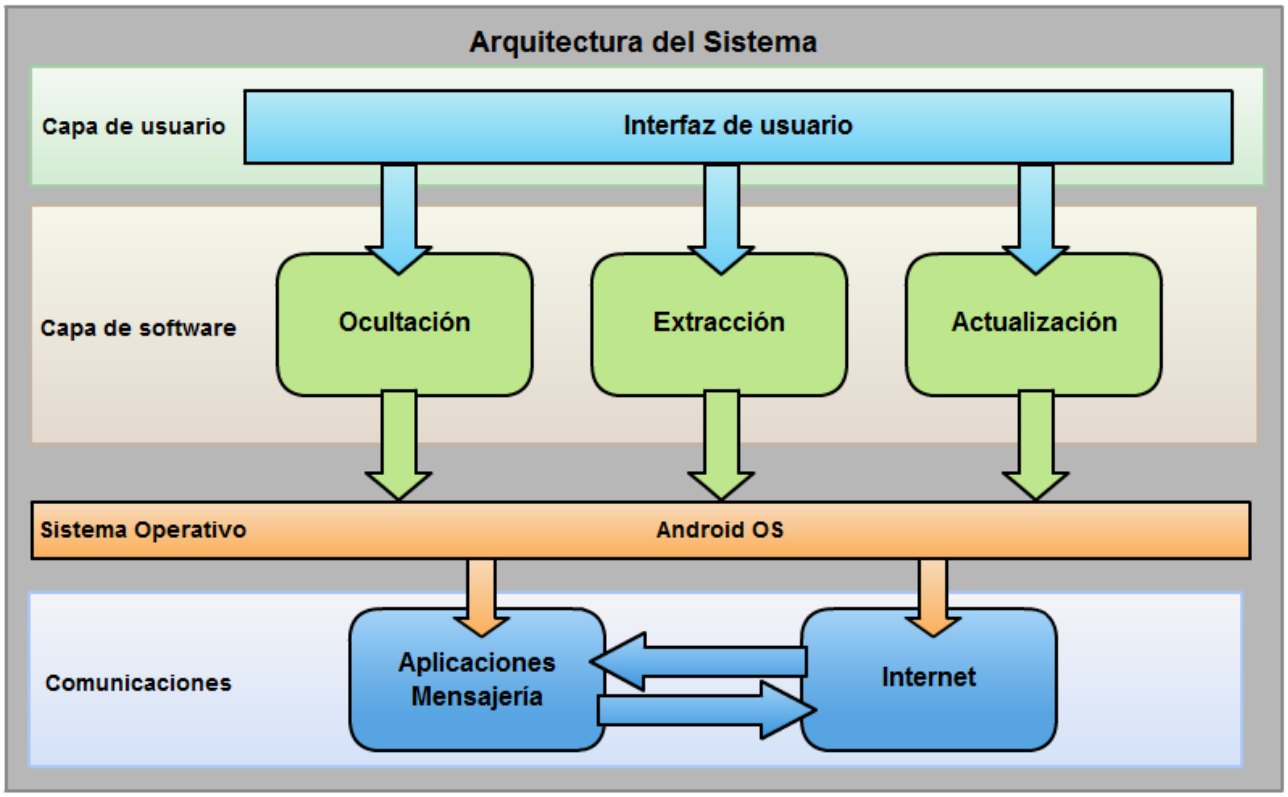


Ilustración 11: Arquitectura del sistema

Se puede observar en la figura 11 que la arquitectura del sistema comienza en la capa de usuario con la interfaz del sistema, la cual se comunica con los 3 módulos que forman la base de operación de la aplicación. A su vez, la aplicación realiza las llamadas al sistema operativo Android oportunas para lograr su objetivo y comunicarse con las aplicaciones de mensajería instantánea. Cabe destacar que es necesario el acceso a Internet por parte de nuestra aplicación para comprobar si hay nuevas actualizaciones, y por parte de los programas de mensajería instantánea para enviar las imágenes que generemos.

A continuación se describen muy brevemente las capas que componen la arquitectura del software a diseñar:

- **La interfaz de usuario:** Es la encargada de comunicar al usuario con la aplicación. Desde ella se podrá acceder a los 3 módulos que conforman el programa, e introducir los datos que el mismo vaya requiriendo para completar su objetivo. Es necesario que esta parte esté bien depurada y sea completamente usable para el usuario.
- **Módulo de ocultación de mensajes:** Partiendo de los parámetros que le suministre el usuario a través de la interfaz, este módulo será capaz de ocultar un mensaje en una imagen seleccionada de manera inapreciable para el ojo humano, para enviarla a través de distintas aplicaciones de mensajería.
- **Módulo de extracción de mensajes:** Será el subsistema encargado de realizar la tarea inversa. Se trata de extraer el mensaje oculto que contuviese una imagen seleccionada por el usuario a partir de una contraseña.
- **Módulo de actualización:** Realiza una tarea bien diferente a la de los módulos anteriores, en este caso comprueba si existen nuevas actualizaciones para la aplicación.
- **Capa de Android:** Los módulos tienen que comunicarse con el sistema operativo que tienen por debajo de la capa de aplicación, en este caso Android, que es el encargado de ejecutar las llamadas al sistema que la aplicación realice.
- **Comunicación externa:** Las imágenes generadas por el programa pueden ser enviadas a otras aplicaciones, para ello será necesario que se establezcan vías de comunicación entre las mismas empleando los mecanismos que Android posee. Asimismo será necesario acceder a Internet para actualizar el sistema si fuera necesario.

El módulo de ocultación y el de extracción forman el pilar esencial del presente proyecto, ya que sin ellos el sistema carecería de utilidad alguna. El módulo de actualización será el último en ser implementado ya que no tiene tanta prioridad como los demás.

5.2 SUBSISTEMAS

Para reducir la complejidad del sistema y facilitar su diseño, éste se ha dividido de forma lógica en subsistemas. La división en subsistemas se ha realizado atendiendo a los requisitos obtenidos durante el análisis y a la arquitectura propuesta en el apartado anterior.

Un subsistema, normalmente, se identifica por los servicios que proporciona, que a su vez conforman un conjunto de funciones con un propósito común. En el caso de la aplicación que se pretende desarrollar, los subsistemas corresponden con cada uno de los módulos definidos en la arquitectura del sistema: **ocultación**, **extracción**, y **actualización**.

Las posteriores secciones estarán dedicadas al diseño de cada uno de los subsistemas anteriores de manera individual, para que sea más sencillo su entendimiento antes de la implementación final del software.

5.3 MÓDULO DE OCULTACIÓN

El módulo de ocultación tiene como objetivo generar una imagen que sea portadora de un mensaje a partir de las especificaciones del usuario. El diseño de este módulo no es precisamente trivial, por ello lo primero que hay que hacer es identificar los componentes y funciones principales que soportan el funcionamiento del subsistema.

En la figura 12 se muestra un diagrama con la estructura principal del módulo de ocultación, se muestran las funciones más importantes que se deben diseñar.

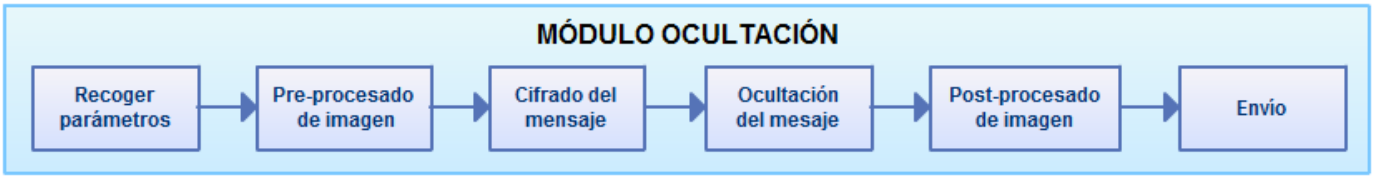


Ilustración 12: Funcionamiento básico del módulo de ocultación

Los 6 componentes que sustentan el modo de ocultación deben ser descritos por separado para entender la funcionalidad completa y detallada del proceso.

5.3.1 RECOGER PARÁMETROS

El usuario se comunica con el programa a través de la pantalla de su dispositivo, desde donde éste puede rellenar los parámetros necesarios para ocultar un mensaje en una imagen.

Es necesario que el usuario introduzca los siguientes datos antes de continuar:

1. Seleccionar la imagen que desea de la galería.
2. Escribir el mensaje que quiere ocultar.
3. Escribir una contraseña para cifrar el mensaje.
4. Elegir si la imagen resultante ha de convertirse a blanco y negro, o a color.
5. Escoger el medio por el que desea enviar la imagen generada.

Mientras el usuario va completando dichos parámetros se validará cada uno de ellos, comprobándose que:

- La ruta especificada para la imagen es correcta y su formato aceptado.
- Todos los campos han sido rellenados.
- No se ha introducido ningún carácter no permitido.

Ante cualquier error que se produzca durante este proceso, el sistema informará al usuario a través de un mensaje emergente.

En la tabla 111 se muestran los parámetros de salida de este proceso, con sus tipos representados en lenguaje JAVA y los posibles valores que pueden tener a nivel interno:

Parámetro	Tipo de dato	Valores
Imagen	Uri	Ruta a imagen
Mensaje	String	1-1000 caracteres
Contraseña	String	Más de un carácter
Modo de color	Boolean	0->B/N 1->Color
Modo de envío	Integer	Número entre 0-4

Tabla 111: Parámetros de salida de este proceso.

5.3.2 PRE-PROCESADO DE IMAGEN

Antes de ocultar la información en la imagen es necesario preparar a ésta. El pre-procesado de imagen se encarga de realizar ciertas operaciones sobre el fichero digital seleccionado por el usuario con el fin de prepararlo para que el resto de procesos que siguen puedan trabajar correctamente. En la figura siguiente (13) se muestra el funcionamiento básico del proceso:

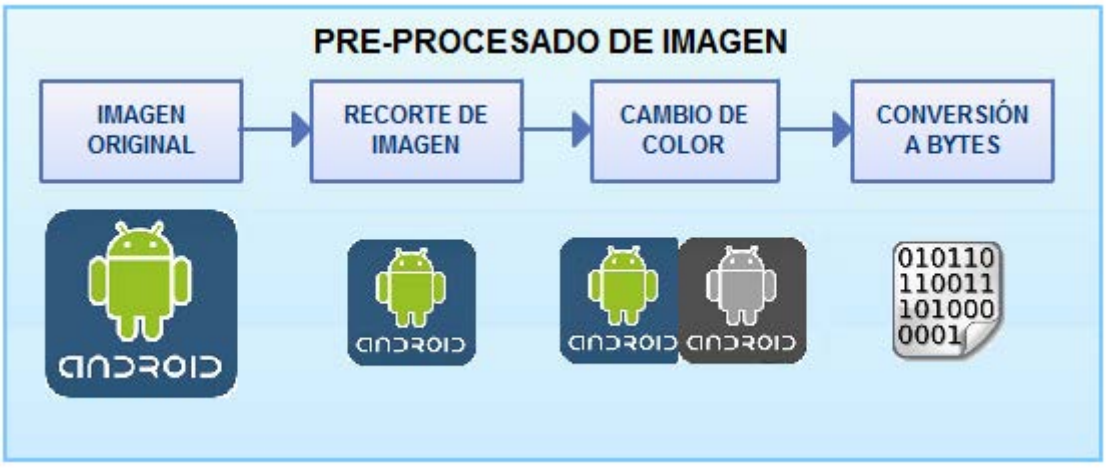


Ilustración 13: Pre-procesado de imagen

Las etapas que conforman el pre-procesado de imagen son las siguientes:

- **Recorte de imagen**: Si la imagen supera las dimensiones permitidas se escala. Esto es especialmente importante para el envío a través de otras aplicaciones, ya que en la mayoría de casos si se superan ciertas dimensiones la imagen se recorta (y pierde el mensaje oculto que transporta).
- **Cambio de color**: Si el usuario ha seleccionado la opción de imagen en blanco y negro la imagen se convertirá a dicho modelo de color tras ser escalada. Por defecto quedará marcado el modo color, con el que no sufrirá modificaciones. Para convertir la imagen a escala de grises en Android basta con reducir la saturación del mapa de bits a cero.
- **Conversión a bytes**: La imagen se convierte en una estructura de bytes, en la que cada píxel queda definido por 3 números enteros que podrán tomar valores desde 0 a 255. Cada uno de esos 3 números que definen el color del píxel corresponden a las componentes del modelo de color RGB, por lo que el primer entero representa la cantidad de “rojo” que tiene el píxel, el segundo la cantidad de “verde, y finalmente el tercer entero compone el color azul. La estructura tendrá la longitud necesaria para albergar cada una de las 3 componentes de cada píxel.

5.3.3 CIFRADO DEL MENSAJE

Este proceso es el encargado de implementar los mecanismos criptográficos necesarios para proteger nuestro mensaje. Se compone de 3 sub-procesos que nos ayudarán a hacer completamente incomprensible la información que queramos transmitir, son los siguientes:

Ilustración 14: Proceso de cifrado del mensaje.

- **Cifrado:** El mensaje secreto se cifra utilizando el algoritmo **AES** con una clave de **256 bits** derivada de la contraseña. La clave de cifrado se obtiene utilizando la función de derivación **PBKDF2** (*Password-Based Key Derivation Function 2*), la cual deriva más de mil veces la contraseña original empleando funciones resumen y algoritmos criptográficos. En la siguiente imagen se ilustra el proceso de cifrado:

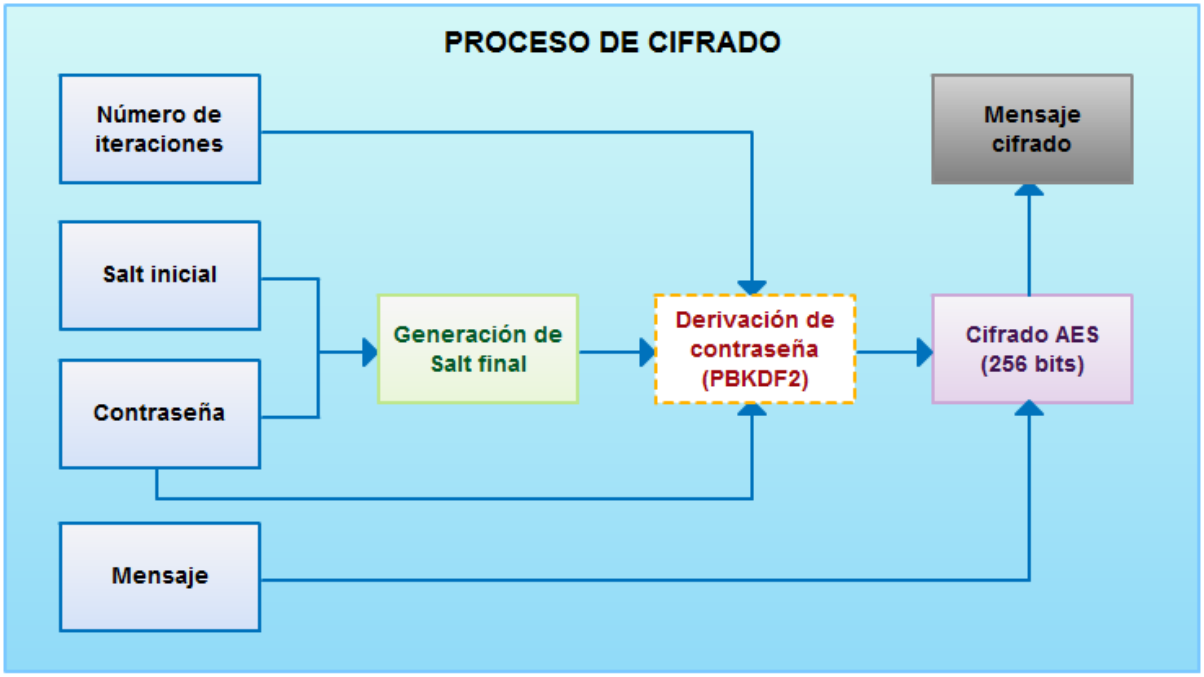


Ilustración 15: Mecanismo de cifrado.

- **Desordenamiento:** El mensaje cifrado se desordena para causar más confusión a un posible atacante. Se ha implementado un algoritmo muy sencillo que sustituye los caracteres de un mensaje, y es completamente reversible. En la figura 16 se muestra un ejemplo de su funcionamiento.
- **Codificación:** El texto cifrado y desordenado es codificado en "***base64***", que permite que el texto resultante sea representado en caracteres *ASCII*, lo cual nos evitará muchos problemas a la hora de introducir la información en la imagen.

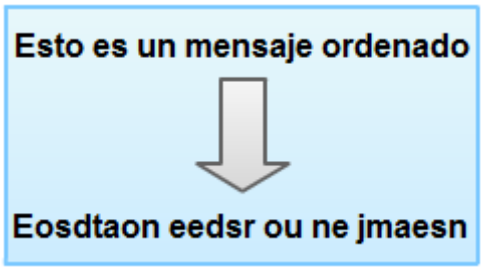


Ilustración 16: Ejemplo de texto desordenado.

5.3.4 OCULTACIÓN DEL MENSAJE

En este proceso va a ser necesario emplear varias técnicas de esteganografía. Se trata sin duda del proceso más complejo del sistema, en él emplearemos técnicas de dispersión, camuflaje y corrección de errores. Para esconder el mensaje cifrado en la imagen se van a llevar a cabo los siguientes pasos:

1. **Obtener la longitud del mensaje:** Necesitamos conocer la longitud del mensaje para insertarla en la imagen, de este modo el receptor sabrá la cantidad de información que hay escondida y podrá recuperarla.
2. **Obtener mapa de dispersión:** Se trata de la tarea más vital del proceso. Se encarga de seleccionar de manera pseudo-aleatoria, basándose en derivaciones de la contraseña de usuario, los bytes de la imagen en los que se va a insertar la información. Es necesario conocer la longitud del mensaje para obtener tantas posiciones de inserción como sean necesarias.
3. **Introducir longitud del mensaje en la imagen:** La longitud es introducida en la imagen utilizando técnicas de sustitución a nivel de bit, que se comentarán en profundidad en el capítulo de implementación.
4. **Introducir mensaje secreto:** El mensaje cifrado se introduce, al igual que la longitud, a lo largo de toda la imagen, en las posiciones que se hayan determinado al generar el mapa de dispersión.
5. **Inserción de redundancia:** Se introducen 3 bits de redundancia adicionales para cada bit de información introducido. De este modo, si se produce un error durante el envío, el sistema podrá examinar los bits de redundancia y comprobar que todos los valores son correctos. En el caso de que alguno quede dañado, este sistema intentará recuperar la información almacenada en los bits de redundancia. En el capítulo de implementación se comentará específicamente el algoritmo empleado y el sistema de corrección de errores.

La implementación de estos métodos será comentada en el capítulo de implementación, en el que se incluirán los algoritmos empleados. Es importante recordar que estos pasos necesitarán desarrollos distintos dependiendo del método de envío que escojamos, así no es lo mismo introducir información en un medio que no va a sufrir compresiones, que hacerlo en otro que sí lo hará.

5.3.5 POST-PROCESADO DE IMAGEN

Esta es la última etapa del módulo de ocultamiento antes de enviar la imagen al programa de mensajería solicitado. Los pasos que deben realizarse en este proceso son:

1. **Convertir la estructura de bytes en una imagen:** Se trata de recoger la estructura en la que se ha insertado el mensaje oculto, y convertirla de nuevo en un fichero de imagen. Para ello será necesario invertir el proceso de descomposición RGB que realizamos en un principio. Cada grupo de 3 bytes de la estructura volverá a formar un píxel de la imagen.
2. **Guardar la imagen:** Una vez se ha formado la imagen a partir de sus bytes, es necesario guardarla en el directorio "Estegano" de la galería para que posteriormente sea enviado. El proceso de guardado de imagen implica una compresión, esta compresión será siempre fijada con el parámetro de máxima calidad y con formato de destino PNG, para que la imagen se guarde sin alterar ninguno de sus píxeles (es una compresión sin pérdida, "*lossless*").

5.3.6 ENVÍO

Esta tarea culmina el proceso enviando la imagen guardada en el proceso anterior. Para enviar la imagen a las distintas aplicaciones se emplearán los sistemas de comunicación entre aplicaciones que proporciona Android, también conocidos como "***Intents***".

Las aplicaciones a las que se puede mandar la imagen son las siguientes:

- **WhatsApp:** Se trata de la aplicación de mensajería instantánea más utilizada en el mundo, cuenta con mayor número de usuarios que sus competidores directos. Permite enviar imágenes, sin embargo aplica una agresiva compresión JPEG sobre las mismas.
- **Line:** Muchos dicen que será la aplicación de mensajería que desbanque a WhatsApp. Desarrollada por unos japoneses, es completamente gratuita, también impone compresión sobre sus imágenes.
- **Spotbros:** La alternativa a las dos anteriores, tiene funciones que lo hace un medio de comunicación instantánea bastante atractivo. Aunque no cuenta con tantos usuarios que lo utilicen, sigue siendo una gran promesa. Al igual que las demás, las imágenes son comprimidas a formato JPEG cuando son enviadas.
- **Email:** Es el medio de comunicación por excelencia en Internet junto con las redes sociales. Por lo general, cuando se envían imágenes adjuntas por correo electrónico, éstas no sufren compresión alguna, por ejemplo *GMail.* Es un medio idóneo para tener comunicaciones secretas.
- **MMS:** Se trata del sistema de mensajería clásico de los dispositivos móviles. Emplea las redes de telefonía para enviar las imágenes adjuntas en el mensaje. La principal limitación es que la imagen no puede exceder de un tamaño límite (aproximadamente 300KB), en el caso de que supere dicho peso será comprimido agresivamente.

Cada programa de destino posee, como hemos visto, características diferentes. Será necesario llevar a cabo en el capítulo de implementación un estudio detallado de cada uno, en el que se aplicarán técnicas de ingeniería inversa para poder obtener los parámetros de compresión que emplean. Con esos parámetros seremos capaces de crear funciones que eviten que el mensaje se dañe en dicho proceso.

5.4 MÓDULO DE EXTRACCIÓN

El módulo de extracción es el encargado invertir el proceso de ocultación, de manera que se pueda recuperar el mensaje que esté oculto en una imagen. El diseño de este módulo es más sencillo que el anterior, en la figura 17 se muestra el diagrama estructural de este subsistema.

Ilustración 17: Funcionamiento básico del módulo de extracción.

A continuación se van a describir los 5 procesos que componen este subsistema.

5.4.1 RECOGER PARÁMETROS

A diferencia del subsistema de ocultación, aquí el usuario sólo tiene que introducir la imagen de la cual quiere extraer el mensaje, y la contraseña con la que lo ocultó. Con estos dos parámetros el sistema es capaz de recuperar el mensaje secreto.

Si el usuario ha introducido una contraseña incorrecta será notificado por pantalla y podrá volver a escribirla de nuevo por si se ha equivocado.

5.4.2 PROCESADO DE IMAGEN

La única tarea que se lleva a cabo en este proceso consiste en la transformación de la imagen en una estructura de bytes. Al igual que en la ocultación, la imagen se divide en bytes que representan el valor de sus píxeles.

No es necesario llevar a cabo ningún procesamiento adicional de la imagen.

5.4.3 EXTRAER MENSAJE

Este proceso es el más complejo del subsistema, se trata de invertir el proceso de ocultamiento, de manera que podamos recuperar el mensaje cifrado que fue introducido. Las funciones más importantes que se llevan a cabo son:

1. **Obtención del mapa de dispersión:** A partir de la contraseña somos capaces de recuperar todas aquellas posiciones de la imagen en las que se ocultó la longitud del mensaje y a él mismo. Igual que en el proceso de ocultación es necesario aplicar diversas transformaciones sobre la contraseña para obtener dicho mapa de posiciones.
2. **Recuperación de la longitud:** Una vez hemos obtenido las posiciones que corresponden a la longitud del mensaje se leen, esta longitud nos indicará el número de posiciones que hay que buscar en el mapa de dispersión para recomponer el mensaje cifrado.
3. **Recuperación del mensaje cifrado:** Se leen tantas posiciones en el mapa de dispersión como indique la longitud, una vez se ha reunido el mensaje se procede al descifrado.

4. **Recuperación de errores:** Cada bit leído cuenta con otros 3 bits adicionales de redundancia situados en otra posición distinta de la imagen. Se comprueba si la redundancia es correcta y, si no lo es, se corrigen los errores. Esta tarea será descrita más profundamente en la sección de implementación.

5.4.4 DESCIFRADO DEL MENSAJE

Tras el proceso de extracción del mensaje obtenemos una cadena de caracteres incomprensibles codificados en ***base64***, será necesario por tanto volver a invertir el proceso de cifrado que se realizó:

1. **Decodificación:** Se invierte el proceso de codificación en base64, ya que es completamente reversible. De este modo obtenemos caracteres no codificados, que están desordenados y cifrados.
2. **Ordenamiento:** Se invoca el proceso de reordenamiento del mensaje, también es totalmente reversible.
3. **Descifrado:** Esta es la etapa clave del proceso, ya que si la contraseña es incorrecta no se podrá obtener el mensaje original. Se utiliza el mismo procedimiento, empleando la misma derivación de la contraseña, el mismo algoritmo AES, pero indicando que se trata del proceso de descifrado.

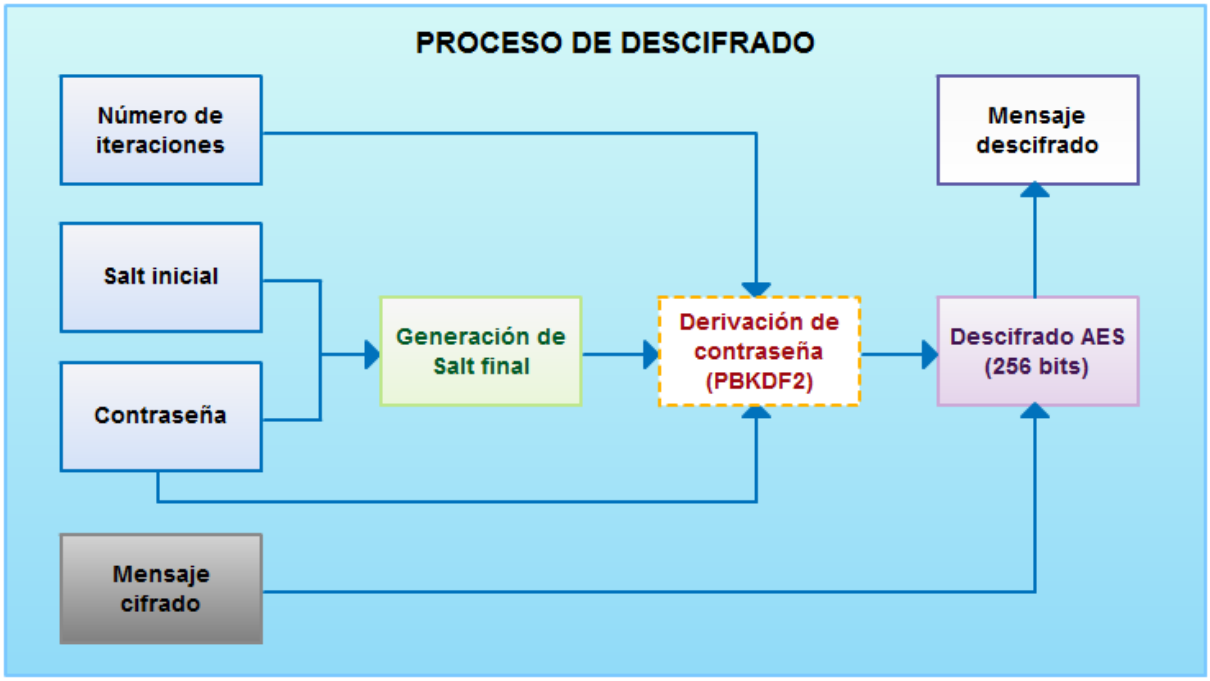


Ilustración 18: Proceso de descifrado.

5.4.5 MOSTRAR MENSAJE

En el caso de que todos los procesos anteriores hayan funcionado correctamente se mostrará al usuario el mensaje descifrado por pantalla.

En caso de que se produzca un error durante cualquier etapa del proceso se mostrará por pantalla un mensaje de error genérico que indicará que la contraseña es inválida. Los errores se pueden producir debido a que:

- La contraseña introducida no es correcta.
- La imagen no contiene ningún mensaje oculto.
- El mensaje oculto está dañado y es imposible recuperarlo.

5.5 MÓDULO DE ACTUALIZACIÓN

El funcionamiento del presente subsistema depende la conectividad a Internet, ya sea por red de datos o por conexión inalámbrica WiFi. El objetivo es verificar si existe alguna actualización nueva del software, y si es así redirigir al sitio apropiado para comenzar con la descarga.

Para comprobar si existen nuevas actualizaciones habrá dos sistemas:

- **Comprobaciones periódicas:** Diariamente, si se dispone de conexión a Internet, se comprobará si existe alguna actualización del software más reciente.
- **Comprobaciones manuales:** El usuario dispondrá de un botón en el menú de la pantalla principal de la aplicación desde el que podrá accionar esta comprobación.

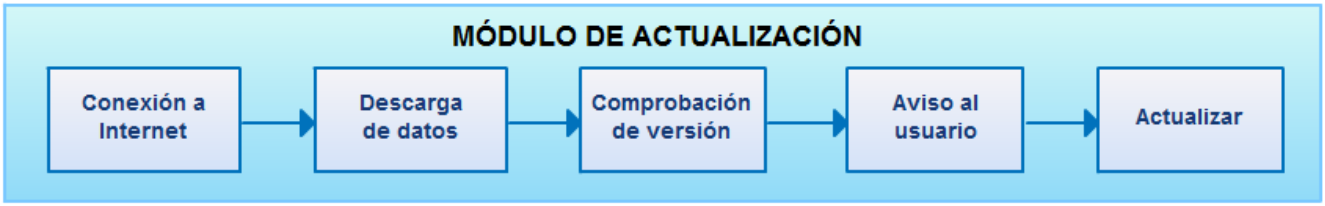


Ilustración 19: Funcionamiento básico del módulo de actualización.

A continuación se describe brevemente la funcionalidad de cada uno de los procesos mostrados en la ilustración anterior:

5.5.1 CONEXIÓN A INTERNET

Se comprobará si el dispositivo está conectado a la red, para ello se realizará una llamada al manejador de conectividad (*ConnectivityManager*) de Android.

En caso de no disponer de conexión se avisará al usuario y finalizará la ejecución del módulo.

5.5.2 DESCARGA DE DATOS

Se realiza una conexión a un servidor situado en ***Dropbox*** para descargar:

- **Un fichero de versión:** Contiene el número de versión acompañado de ciertos parámetros, que especifican el modo en que debe realizarse la actualización.
- **Un fichero de notas:** En caso de que la versión de la aplicación sea más antigua que la leída en el fichero de versión, se descargará este fichero adicional que mostrará por pantalla al usuario el mensaje que contenga. Por ejemplo, se pueden mostrar las novedades de la nueva versión, una advertencia de la necesidad de la actualización, etc.

5.5.3 COMPROBACIÓN DE VERSIÓN

Tras haber efectuado la descarga del fichero de versión desde *Dropbox*, se comprobará el contenido del mismo, que puede contener la siguiente información:

- **Un número** que representa la última revisión de la aplicación disponible. La primera versión de la aplicación corresponde al número 1, la segunda al 2, y así sucesivamente, sólo se aceptan números enteros. Este identificador es obligatorio.
- Adicionalmente puede contener **una o más letras,** que indican a la aplicación si necesita realizar alguna acción adicional para descargar la nueva versión. Las letras, si las hubiese, van concatenadas tras el número de versión y del carácter "**:**". Las posibles letras que se pueden encontrar son:
    - **D:** Si este identificador está presente en el fichero de versión se está indicando al programa que la actualización la debe realizar a través del repositorio situado en *Dropbox.* Esto puede deberse a que el repositorio de descarga principal no está disponible.
    - **N:** Indica a la aplicación que debe mostrar el contenido del fichero de notas al usuario por pantalla.
    - **F:** En caso de que la actualización sea obligatoria este identificador estará presente. Exige al usuario a que actualice a la última versión para poder continuar utilizando la aplicación, en caso de que éste rechace la proposición el programa se cerrará. Esta opción es especialmente útil si se ha producido un error muy grave con cierta versión de la aplicación y el usuario pudiera estar expuesto a algún tipo de vulnerabilidad o daño.

  Por ejemplo, si el archivo de versión contuviera "**4:DN**" se estaría indicando a la aplicación que la última versión disponible es la 4 y, en caso de que esta fuera superior de la actual, que muestre por pantalla al usuario el contenido del fichero de notas. Adicionalmente se indica que la descarga del fichero ha de efectuarse desde el servidor de *Dropbox*.

5.5.4 AVISO AL USUARIO

En caso de que la versión actual sea la misma que la última disponible se informará al usuario a través de un mensaje emergente. Asimismo, si la sintaxis del fichero de versión es incorrecta no se procederá con la descarga de la nueva versión.

Cuan haya una actualización disponible se mostrará al usuario por pantalla un cuadro de diálogo desde el que puede aceptar o rechazar la descarga de la nueva versión del sistema.

5.5.5 ACTUALIZAR

Si el usuario acepta la descarga del nuevo software, se comprobará si la descarga debe efectuarse desde el sitio predeterminado o desde el repositorio de *Dropbox*, para ello comprobará el identificador "**D**" del archivo de versión.

A continuación se muestra el contenido de la carpeta de *Dropbox* desde la que se descargará el archivo de versión, el de notas, y eventualmente la aplicación. Nótese que "***Hide&Seek***" es el nombre de la aplicación que se ha desarrollado.

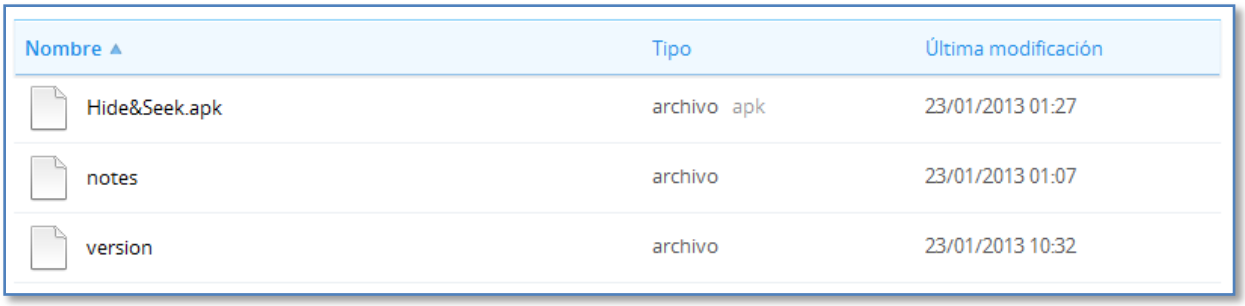


Ilustración 20: Repositorio de actualizaciones en Dropbox.

5.6 INTERFACES DE USUARIO

A continuación se especifica el diseño que deben presentar las distintas interfaces del sistema.

5.6.1 PANTALLA PRINCIPAL

La pantalla principal de la interfaz presenta los 2 botones principales para acceder a los modos de ocultación y extracción, se les ha llamado "**Ocultar**" y "**Descifrar**" respectivamente para facilitar la comprensión por parte del usuario. En la parte inferior de la pantalla se muestra el menú contextual de la aplicación, en el que se observan 2 opciones: la primera "**Actualizar**" para comprobar si hay nuevas actualizaciones disponibles; la segunda "**Acerca de**" muestra información sobre la versión de la aplicación, el autor, y un correo electrónico de contacto.

Ilustración 21: Pantalla principal de la aplicación.

Esta interfaz se caracteriza por su sencillez, pues una de las características principales que debe tener el programa es la facilidad de uso. En la barra superior de la aplicación se muestra el nombre del software "**Hide&Seek**" ("esconder y encontrar" del inglés).

5.6.2 MODO OCULTACIÓN

Para que el subsistema pueda ocultar mensajes secretos en imágenes es necesario que el usuario introduzca por pantalla los parámetros requeridos por la aplicación, tal y como se explicó anteriormente.

Se ha decidido seguir un diseño en forma de "**asistente**", en el que la interfaz interactuará con el usuario pidiéndole paso a paso que complete la información. Para avanzar a la hora de rellenar los campos se ha creado un botón "**Siguiente**", que cuando sea pulsado comprobará si se ha rellenado el campo solicitado y verificará que los datos introducidos son correctos.

Ilustración 22: Pantalla selección Imagen.

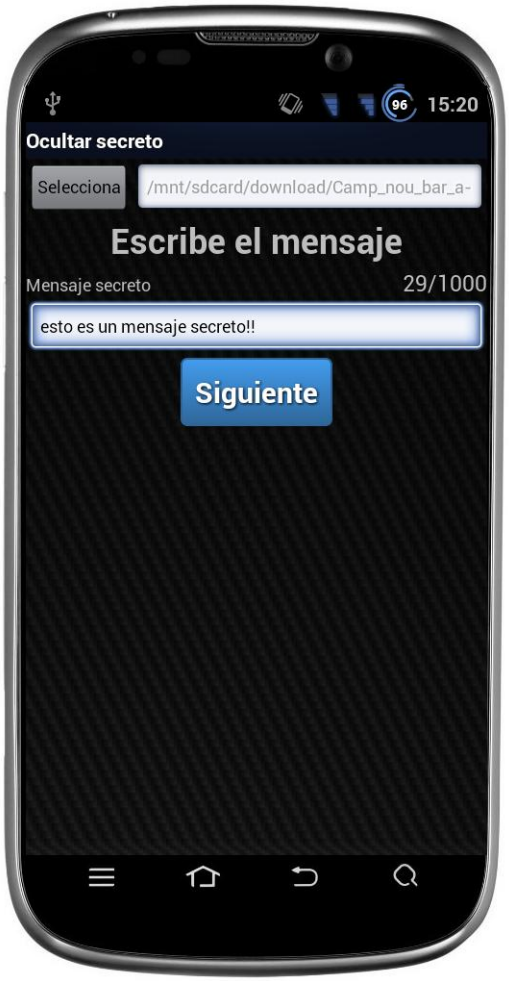


Ilustración 23: Pantalla escritura mensaje.

En la figura 22 la pantalla muestra un botón "**Selecciona**" que redirige al usuario hasta la galería, donde puede seleccionar la imagen preferida. Una vez seleccionada la imagen aparecerá su ruta en el campo de texto situado a su derecha. Si la imagen existe el usuario podrá pulsar el botón "Siguiente" para avanzar.

En la figura 23 se muestra cómo la interfaz pide al usuario que introduzca un mensaje de texto. Encima del campo de texto a la derecha se muestra un contador de caracteres, que nos indica en todo momento cuantos caracteres llevamos escritos.

Ilustración 24: Pantalla inserción contraseña.

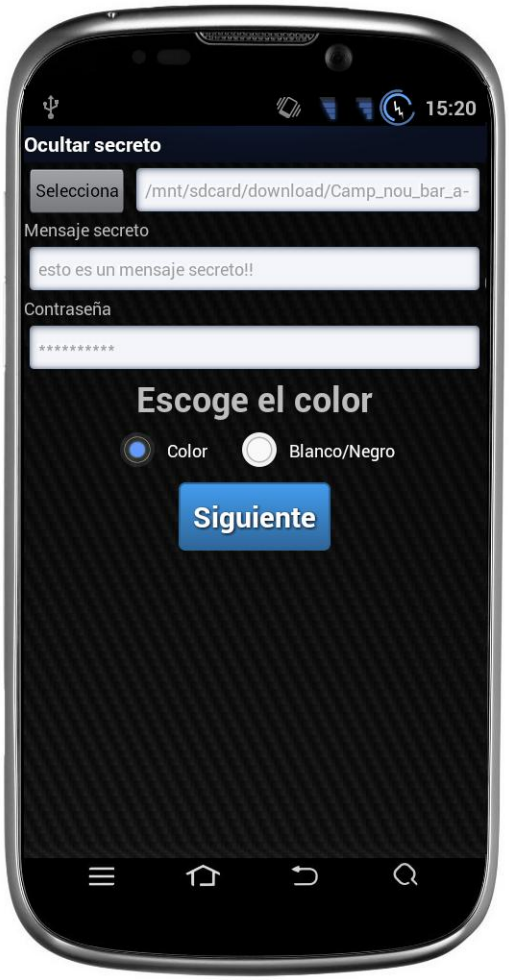


Ilustración 25: Pantalla selección de color.

La ilustración 24 muestra cómo la aplicación nos pide que insertemos una contraseña, que será posteriormente la encargada de cifrar el mensaje y seleccionar el mapa de dispersión en la imagen. Se puede observar que la contraseña no es ocultada con asteriscos inmediatamente, en su lugar se muestra al usuario para que este se cerciore de que la ha introducido bien. En el momento que pulsemos el botón “Siguiente” la contraseña será ocultada por asteriscos que impedirán que sea visible.

El siguiente paso en el proceso de recolección de parámetros es la elección del modelo de color, como se muestra en la figura 25. Por defecto estará marcado el modelo de color convencional, que es en la mayoría de casos el más empleado por los usuarios. Para la selección se ha empleado un “botón de elección” (*radio-button)*, que marcará en azul la opción seleccionada por el usuario.

Se observa como los campos ya completados quedan sombreados en gris, de manera que para editarlos sería necesario pulsar el botón “**atrás**” de Android (el tercero de los mostrados en la parte inferior del dispositivo de la imagen).

Ilustración 26: Pantalla selección de modo de envío.

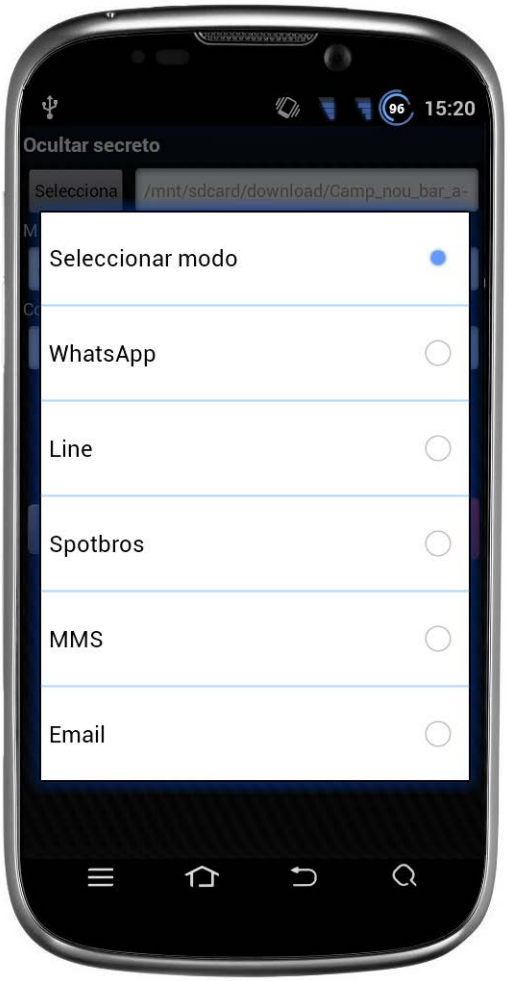


Ilustración 27: Desplegable modo de envío.

Tras haber escogido el color de la imagen, la interfaz nos muestra un listado deplegable para seleccionar el programa de destino con el que deseamos enviar la imagen portadora del mensaje. En la figura 26, se muestra dicho listado acompañado de un botón de "**Ayuda**", cuya función consiste en indicar al usuario la utilidad de cada modo de envío, por si tiene dudas.

La ilustración 27 muestra las opciones que se pueden elegir para enviar nuestra imagen. Es importante resaltar que sólo se puede marcar una de ellas. En el caso de que el usuario no tenga instaladas una o varias de las aplicaciones mostradas, el listado no mostrará dichas aplicaciones.

En el momento que el usuario haya completado todos los campos anteriormente descrito podrá pulsar el botón inferior "**Ocultar**" para proceder con la inserción del mensaje oculto en la imagen seleccionada.

Ilustración 28: Diálogo de progreso de ocultación.

Ilustración 29: Ejemplo de envío por WhatsApp.

Mientras se cifra y oculta la información, entre otros procesos, la interfaz mostrará por pantalla el progreso real de la ejecución. Puede darse el caso de que la barra de progreso se reinicie durante el proceso de ocultación, esto ocurre cuando el sistema no ha sido capaz de ocultar la información con los parámetros establecidos, con lo cual reintenta el proceso de nuevo pero reduciendo la saturación de la imagen, lo que facilita la inserción del mensaje.

En el peor de los casos, si la aplicación ha sido incapaz de insertar el mensaje en 5 reintentos, se mostrará un mensaje por pantalla al usuario indicándoselo. En la mayoría de casos esto sucede debido a que se intenta insertar mucha información en una imagen, que por sus características, es inadecuada para ocultar datos.

Cuando se ha completado el proceso de ocultación, se redirigirá al usuario a la pantalla de la aplicación que haya seleccionado como destino para enviar la imagen. En la figura 28 observamos, en el caso de haber seleccionado la opción de WhatsApp, que podemos enviar la imagen portadora a cualquiera de los contactos que tengamos en la aplicación de mensajería. La imagen sufrirá compresión en este caso, pero nuestro mensaje oculto en la imagen sobrevivirá y llegará en buen estado al receptor.

5.6.3 MODO DE EXTRACCIÓN

El modo de extracción funciona de manera muy similar al de ocultación, la principal diferencia radica en que sólo se nos pedirá la introducción de 2 parámetros por pantalla: la imagen y la contraseña.

Existen dos maneras distintas de abrir una foto para extraer su mensaje oculto: la primera es desde la interfaz principal de la aplicación, en la que al igual que en el modo de ocultación disponemos de un botón para seleccionar de la galería la foto a emplear; y la segunda opción, la más cómoda en algunos casos, consiste en abrir la imagen directamente desde el menú "**Compartir con**" o "**Enviar a**" de Android y seleccionar la opción "**Encontrar secreto**" como se muestra en la figura 31.

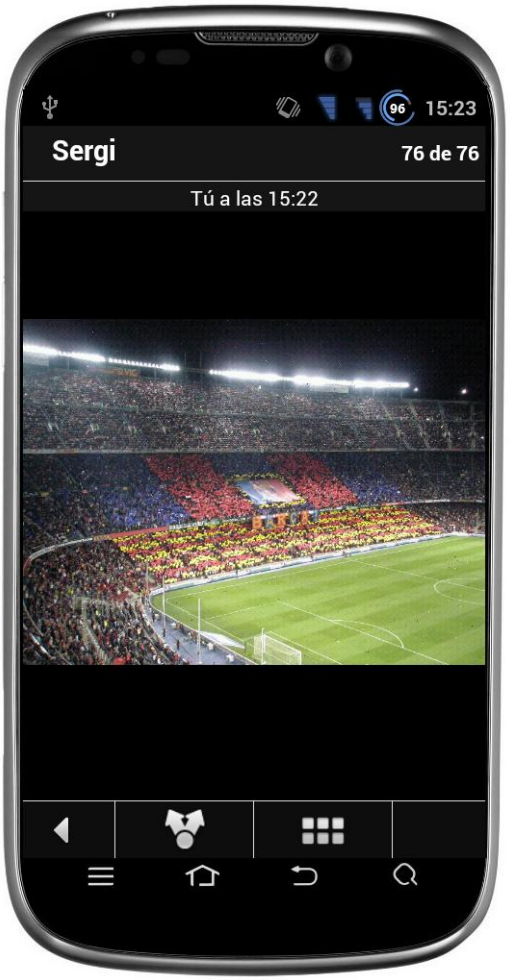


Ilustración 30: Pantalla de imagen abierta.

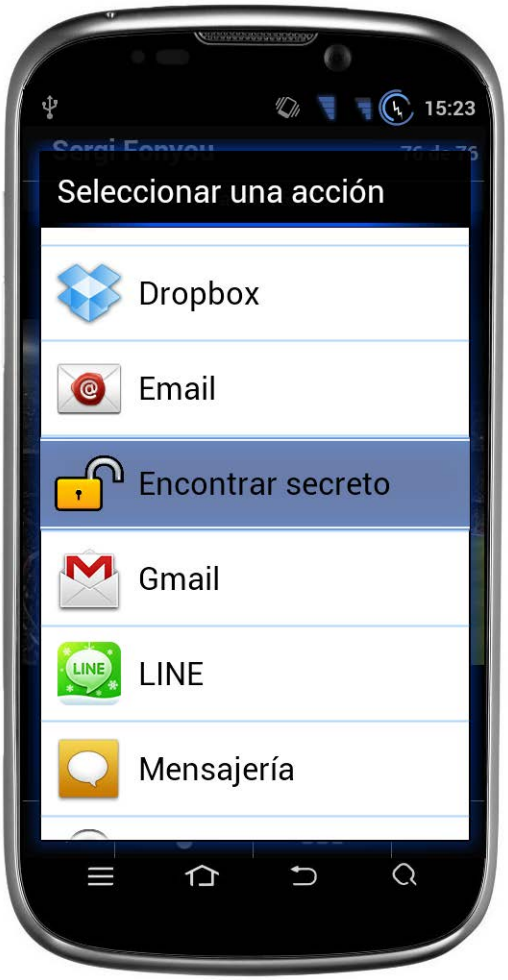


Ilustración 31: Selección de opción "Compartir con".

La figura 30 muestra una imagen abierta desde WhatsApp. En la parte inferior de la misma hay dos botones, el primero de ellos es la opción "Compartir con" de Android, que nos permite enviar dicha imagen a otras aplicaciones como se muestra en la figura 31, en la que hemos de seleccionar la opción que tiene el icono del"candado abierto" para abrir la imagen con la aplicación.

En el momento que seleccionemos la presente aplicación desde la lista de aplicaciones mostradas, se enviará la ruta en la que está guardada la imagen al modo de extracción de mensajes, el cual completará automáticamente el primer paso y rellenará el campo de ruta con la recibida, como se muestra en la ilustración 32.

Ilustración 32: Pantalla de selección de imagen 2.

Ilustración 33: Inserción de contraseña 2.

Ahora que se ha seleccionado la imagen tenemos que introducir la contraseña con la que se ocultó el mensaje secreto. Se supone que emisor y receptor conocen dicha contraseña, sin la cual sería imposible recuperar la información original.

La figura 33 muestra la pantalla previa a la recuperación del mensaje, tras introducir la contrasñea se debe pulsar el botón "**Descifrar**" presente en la parte inferior del cuadro de texto.

La aplicación intentará rescatar la información del interior de la imagen, mientras tanto se mostrará un diálogo de espera por pantalla que indica al usuario que se está intentando recuperar el mensaje, como se observa en la figura 34.

Si el proceso ha funcionado y la contraseña era la correcta se mostrará por pantalla, como indica la figura 35, el mensaje secreto.

En caso de que no haya sido posible recuperar el mensaje se indicará al usuario que la contraseña era incorrecta, y se le permite reintroducirla de nuevo.

Ilustración 34: Diálogo de extracción.

Ilustración 35: Pantalla de mensaje extraído.

6 IMPLEMENTACIÓN

Este capítulo aborda la implementación del sistema de información basándose en el análisis y diseño del mismo realizados en los capítulos anteriores.

6.1 INGENIERÍA INVERSA

En esta sección se emplearán técnicas de ingeniería inversa sobre los siguientes programas:

- **WhatsApp.**
- **Line.**
- **Spotbros.**
- **MMS (mensajería multimedia).**

Todo ello será necesario con el objetivo de obtener los detalles internos de la implementación de cada una de estas aplicaciones. La información que se pretende extraer de cada una de ellas es:

1. Obtener los **parámetros de compresión** utilizados en cada caso para el envío de imágenes a través de sus protocolos.
2. Obtener los detalles de implementación para el **escalado de imágenes**.
3. Encontrar los mecanismos necesarios para lograr una **comunicación** directa con dichas aplicaciones, con el objetivo de enviar imágenes a sus respectivas interfaces.

6.1.1 WHATSAPP

En primer lugar, para realizar un proceso de ingeniería inversa sobre un programa es necesario disponer de él, por ello se ha descargado la última versión de WhatsApp en el teléfono móvil. Posteriormente se ha exportado a formato **APK** para trabajar con él desde Linux.

La versión que se ha empleado en este proceso es la 2.9.378. Empleando el programa **dex2jar** se ha conseguido exportar el archivo del programa a formato **JAR**, que es una de las extensiones de los ficheros JAVA ejecutables. Finalmente empleando la utilidad **J-GUI**, un descompilador para JAVA, se han podido obtener los ficheros JAVA de WhatsApp.

Se ha podido comprobar que el código descompilado está bastante ofuscado, es decir, es difícil comprender su funcionamiento. Hay herramientas para Android como ***ProGuard*** que realizan este tipo de ofuscación antes de realizar la compilación final del programa, normalmente esto se hace para evitar la ingeniería inversa de las aplicaciones.

COMPRESIÓN

Para hallar la compresión que impone WhatsApp sobre las imágenes se ha buscado en el código fuente descompilado el método de compresión específico que emplea Android para comprimir una imagen.

El método de compresión que usa Android es el siguiente:

```
public boolean compress (Bitmap.CompressFormat format, int quality, OutputStream stream)
```

Donde los atributos que nos interesan son "**format**", que indica el formato de imagen resultante, y "**quality**", que representa el ratio de compresión a aplicar.

Por tanto buscaremos dicho método en el código fuente del programa, en la figura 36 podemos observar los resultados obtenidos de la búsqueda desde la terminal de Linux.

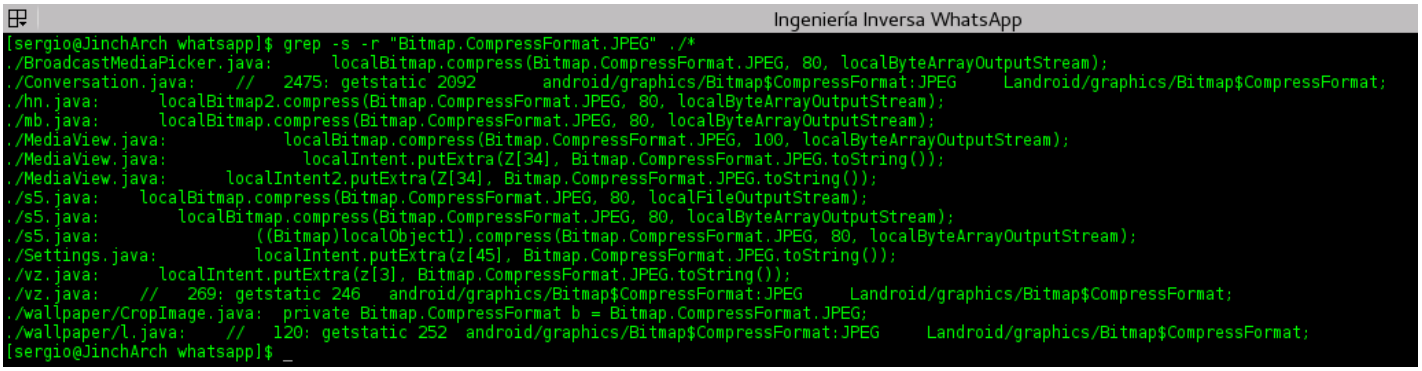


Ilustración 36: Compresión WhatsApp 1.

Observamos que el formato de compresión es el JPEG y que la mayoría de llamadas al método de compresión emplean el parámetro de calidad 80, lo cual es una compresión bastante agresiva.

Para asegurarnos de que 80 es el ratio de compresión empleado se han verificado las clases que han arrojado los resultados anteriores. Se ha averiguado, a pesar de la ofuscación del código, que la clase encargada de la compresión de las imágenes cuando se envían por WhatsApp es "***BroadcastMediaPicker***" (la primera que aparece en la figura 36). Esta clase se encarga de recoger los archivos multimedia que se desean enviar a través de esta aplicación y, en nuestro caso, también se encarga de la compresión de dichos medios.

ESCALADO DE IMÁGENES

Para entender cómo recorta WhatsApp las imágenes se ha buscado en el código fuente uno de los métodos más usuales que emplea Android para escalar imágenes:

```
Bitmap createScaledBitmap (Bitmap src, int dstWidth, int dstHeight, boolean filter)
```

Del anterior método nos interesan los parámetros "***dstWidth***" y "***dstHeight***", que indican respectivamente el ancho y el alto deseado para la imagen resultante a partir de la original.

Los resultados obtenidos de la búsqueda son los mostrados en la figura 37:

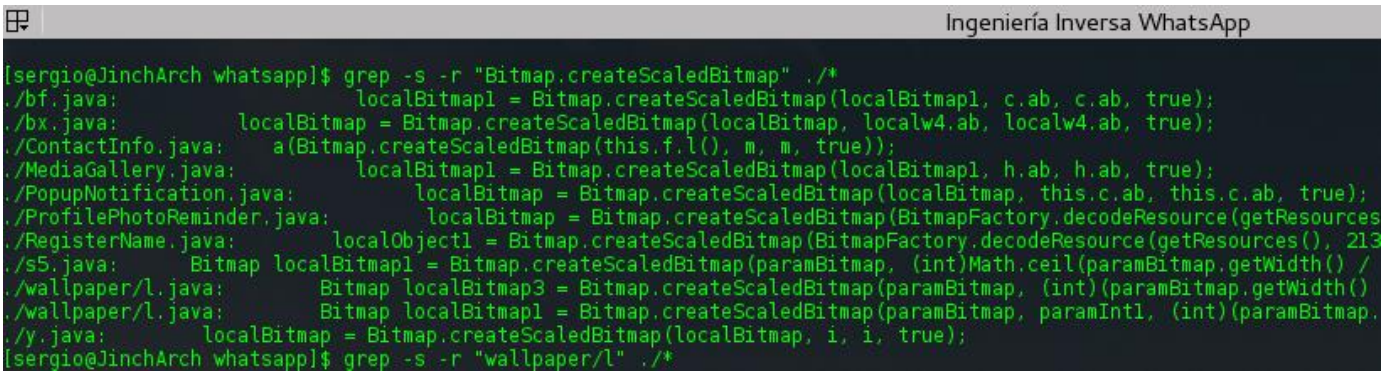


Ilustración 37: Escala de imágenes en WhatsApp.

Aparecen muchas clases que emplean dicho método, sin embargo, se ha analizado con detenimiento cada una de ellas y se ha encontrado que la clase responsable del escalado de las imágenes que se envían es "**wallpapper/l.java**". De esta clase se ha extraído un pequeño recorte del fragmento de código mostrado en la figura 38.

```
float f1 = paramBitmap.getWidth() / paramInt1;
float f2 = paramBitmap.getHeight() / paramInt2;
if (f1 > f2)
{
  Bitmap localBitmap3 = Bitmap.createScaledBitmap(paramBitmap, (int)(paramBitmap.getWidth() / f2), paramInt2, true);
  localBitmap2 = Bitmap.createBitmap(localBitmap3, (localBitmap3.getWidth() - paramInt1) / 2, 0, paramInt1, paramInt2);
  if (localBitmap2 != localBitmap3)
    localBitmap3.recycle();
}
else
{
```

Ilustración 38: Método de escalado de imágenes en WhatsApp.

Se aprecia que el escalado de imagen se realiza a través de unos parámetros prefijados por la aplicación (***paramInt1*** y ***paramInt2***).

No se ha podido determinar el valor exacto de ambos parámetros por lo que se ha recurrido a otra técnica, se trata de analizar los ***logs*** que WhatsApp arroja en su fichero de registro de eventos situado en "*data/data/com.whatsapp/files/Logs/whatsapp.log*". Para acceder a dicha ruta ha sido necesario emplear la terminal de Linux y el comando "***adb shell***", el cual nos permite navegar y efectuar operaciones en el entorno del dispositivo móvil conectado.

Investigando dicho archivo se ha llegado a la siguiente conclusión respecto a los tamaños de escalado:

- El archivo de máximo tamaño que no es escalado tiene una resolución de 1000x1000 píxeles. Por lo tanto, las imágenes que tengan una resolución igual o inferior a esas no será escalado.
- Si el archivo tiene una longitud mayor se intentará escalar proporcionalmente para que las dimensiones finales sean de 800x600 o 600x800.

COMUNICACIÓN CON WHATSAPP

Nuestra aplicación tiene que ser capaz de enviar las imágenes generadas a WhatsApp, para que desde éste se puedan enviar al destinatario que escojamos.

En Android, las aplicaciones se comunican utilizando un mecanismo llamado ***Intent***, que es capaz de realizar llamadas a los componentes de un programa para enviar y recibir información y datos. Cuando se lanza un *Intent* se pueden configurar diversas opciones como son:

- La acción que va a realizar, en nuestro caso la de enviar.
- La información que se va a adjuntar, esto incluye rutas internas a imágenes.
- El tipo de información que se va a enviar, en nuestro caso una imagen.
- Y, en último lugar, el nombre del programa y del componente específico al que deseamos enviar dicha información.

Para obtener esta información se han investigado las llamadas que realizan otras aplicaciones para enviar una imagen a WhatsApp. Se ha empleado el comando "***adb logcat***" para visualizar todos los eventos que se produzcan a nivel de sistema operativo en Android, de modo que en el momento que lancemos una petición desde, por ejemplo, la galería para enviar una imagen a WhatsApp se quedará registrado el tipo de llamada que se ha empleado.

En la figura siguiente se muestra el resultado al realizar dicha petición en tiempo de ejecución:

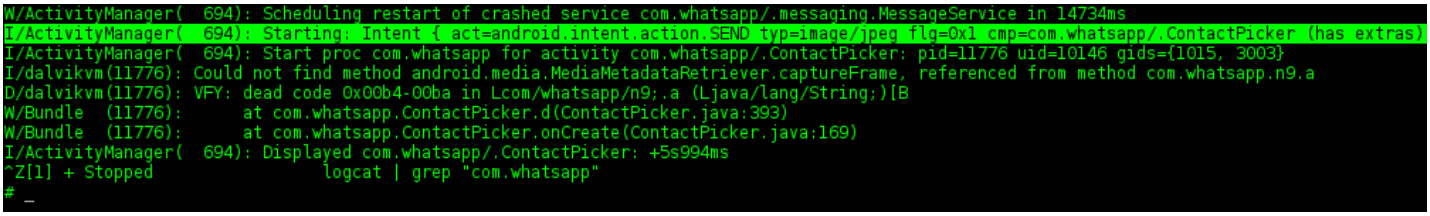


Ilustración 39: Llamada a WhatsApp desde galería.

En la ilustración anterior se destaca la línea que representa la llamada (*Intent)* a WhatsApp, en ella se aprecia el nombre del componente al que se tiene que referenciar en la llamada (***com.whatsapp.ContactPicker)***.

De esta manera se ha podido incorporar en la aplicación que se desarrolla en el presente proyecto la vía de comunicación con la aplicación más famosa de mensajería instantánea del mundo.

6.1.2 LINE

Una vez hemos visto cómo obtener los parámetros de compresión y de comunicación para WhatsApp, se realizará el mismo proceso para la aplicación Line. En el caso del escalado de imágenes se ha comprobado que el funcionamiento es idéntico en Line, por lo que no se describe de nuevo.

COMPRESIÓN

En la siguiente figura vemos una captura de la terminal de Linux desde la que se ha lanzado un comando para que busque entre el código fuente de Line los lugares donde se emplea la compresión.

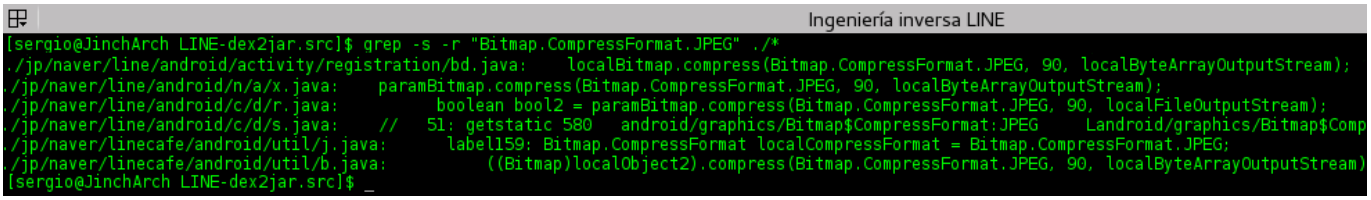


Ilustración 40: Compresión en Line.

Se puede asegurar que el parámetro de compresión que utiliza Line sobre las imágenes es **90**, ya que no aparece otro distinto en todas las llamadas a las funciones de compresión.

COMUNICACIÓN

De la misma manera que con WhatsApp, se ha empleado el visor de eventos de Android (***logcat***) para capturar el *Intent* que necesita nuestra aplicación para enviar imágenes a Line.

En la siguiente captura observamos el momento en el que se realiza la llamada que necesitamos:

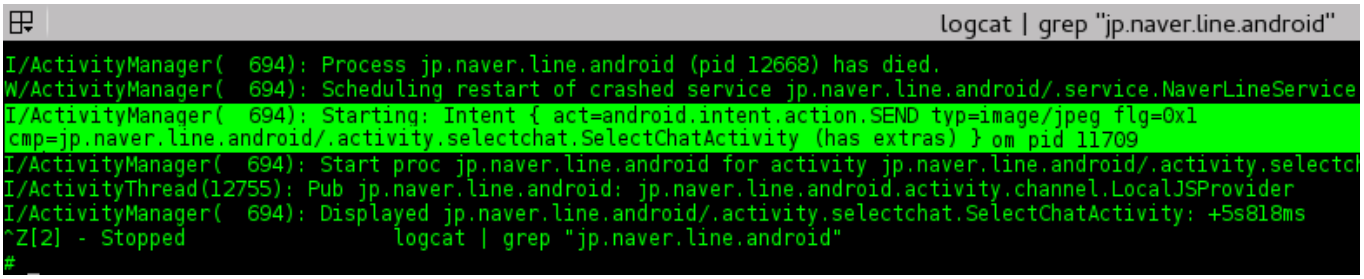


Ilustración 41: Llamada a Line desde la galería.

De la figura 41 podemos inferir que el componente al que hay que realizar la llamada es "***jp.naver.line.android.activity.selectchat.SelectChatActivity***", como se destaca en la línea iluminada en verde.

6.1.3 SPOTBROS

Spotbros trabaja de una manera similar a las aplicaciones anteriores, así que seguiremos el mismo procedimiento para obtener los parámetros que necesitamos en nuestra aplicación.

COMPRESIÓN

En el caso de este programa no ha sido necesario realizar una búsqueda en profundidad en todo el código fuente, se ha identificado el método que realiza la compresión y se muestra en la figura 42.

```
152  private void CompressFile(Bitmap paramBitmap, File paramFile, int paramInt)
153  {
154    try
155    {
156      localFileOutputStream = new FileOutputStream(paramFile);
157      int i = 100 - paramInt * 5;
158      paramBitmap.compress(Bitmap.CompressFormat.JPEG, i, localFileOutputStream);
159    }
160    catch (FileNotFoundException localFileNotFoundException)
161    {
```

Ilustración 42: Método de compresión de Spotbros.

A diferencia de WhatsApp y Line, Spotbros no comprime siempre empleando el mismo valor como atributo de calidad. Como se observa en la en la línea destacada de la figura 42, la asignación de la calidad (variable "i") depende del parámetro "*paramInt*", que varia durante la ejecución del programa entre los valores 0 y 2. Esto implica que el parámetro de calidad que recibe la función de compresión puede ser 100, 95 o 90.

A pesar de no haber podido determinar el valor exacto para la compresión, se han realizado numerosas pruebas con los 3 valores obtenidos y en la mayoría de los casos el valor **90** funciona mejor que el resto. Por tanto este será el parámetro que emplearemos en nuestro sistema a la hora de enviar imágenes por Spotbros.

COMUNICACIÓN

Para determinar la llamada específica para enviar información a Spotbros, se ha empleado una vez más la utilidad *logcat* de Android.

```
                                                    logcat | grep "com.spotbros.activities"
com.spotbros.activities
# logcat | grep "com.spotbros.activities"
I/ActivityManager(  694): Starting: Intent { act=android.intent.action.SEND typ=image/jpeg flg=0x1
cmp=com.spotbros.activities/.MainActivity (has extras) } from pid 11709
W/ActivityManager(  694): Activity pause timeout for HistoryRecord{409d3710 com.spotbros.activities/.MainActivity}
W/ActivityManager(  694): Trying to launch com.spotbros.activities/.ContactsActivity
I/ActivityManager(  694): Displayed com.spotbros.activities/.ContactsActivity: +1s698ms (total +3s220ms)
^Z[3]   Stopped                 logcat | grep "com.spotbros.activities"
# _
```

Ilustración 43: Llamada a Spotbros desde galería.

Como se muestra en la imagen, el componente al que debemos efectuar la llamada para enviar una imagen en Spotbros es "***com.spotbros.activities.MainActivity***", como se muestra en la figura 43.

6.1.4 MMS

En el caso de la aplicación de envío de mensajes multimedia no ha sido necesario llevar a cabo los modos de operación realizados anteriormente. Este programa a diferencia de los demás, al ser nativo de Android, es de código abierto, con lo cual no tendremos que lidiar con descompiladores de JAVA ni líneas de código ofuscadas.

COMPRESIÓN

Examinando el código fuente disponible en Internet [24] de dicha aplicación, se ha encontrado fácilmente el parámetro de compresión para las imágenes, éste puede ser 95 o 50. Se trata de un rango demasiado dispar, por lo que hay que descartar esta vía.

A pesar de ello, existe una vía por la que se puede evitar que los mensajes sean comprimidos. Examinando detenidamente el código fuente se ha descubierto que la aplicación emplea unos parámetros de configuración para regir su funcionamiento (***MmsConfig.java***), en ellos se especifican detalles como el tamaño máximo que puede tener la imagen para no ser comprimida. En la siguiente ilustración se muestra un recorte de dicho fichero, en el que se muestran éste y otros parámetros interesantes.

```
36. public class MmsConfig {
37.     private static final int MAX_IMAGE_HEIGHT = 480;
38.     private static final int MAX_IMAGE_WIDTH = 640;
39.     private static final int MAX_TEXT_LENGTH = 2000;
40.     private static int mMaxMessageSize = 300 * 1024;  // default to 300k max size
```

Ilustración 44: Parámetros compresión MMS [24].

En la figura 44 se muestra que:

- La máxima altura que puede tener la imagen es de 480 píxeles.
- La máxima anchura que puede tener la imagen es de 640 píxeles.
- El tamaño del mensaje no puede superar los **300KB** de peso.

Es importante destacar que una imagen de 480x640, resoluciones máximas permitidas, ocupa el mismo tamaño que el máximo peso que puede tener el mensaje completo, 300KB.

Conociendo el tamaño que puede tener la imagen multimedia en el caso más extremo (300KB), podemos seguir el siguiente procedimiento para evitar que nuestros datos introducidos en la imagen se pierdan:

1. Comprimir la imagen que se quiere enviar hasta que pese menos de 300KB.
2. Introducir el mensaje secreto dentro de la imagen.
3. Enviarla por MMS sin riesgo de que la imagen sea comprimida y se pierda el mensaje secreto.

Para la implementación real, se ha establecido de tamaño máximo para la imagen un tope de **220KB** antes de insertar el mensaje en su interior, de este modo nos aseguramos que en ningún caso la imagen supere el tamaño límite a partir del cual se aplicaría compresión.

Se trata de una manera elegante de resolver el problema que los mensajes multimedia suponen. Para la reducción de tamaño se ha empleado una función de escalado similar a la que aplican las aplicaciones de mensajería instantánea.

COMUNICACIÓN

Para llamar al componente que se encarga de adjuntar una imagen a un mensaje multimedia se ha empleado de nuevo la herramienta ***logcat*** de Android:

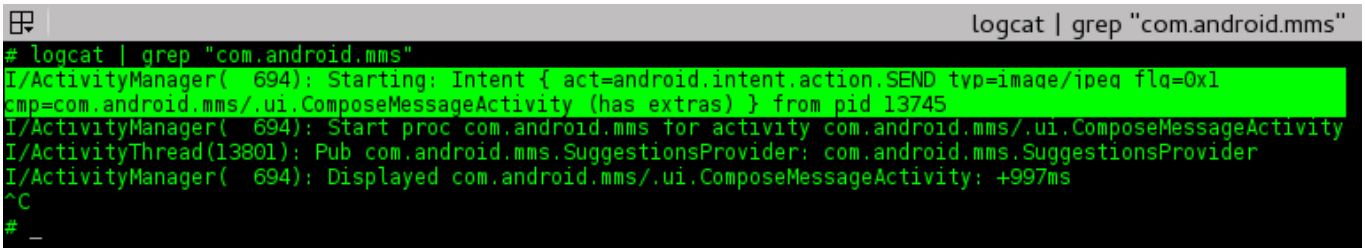


Ilustración 45: Llamada a MMS desde galería.

El componente de comunicación es "***com.android.mms.ui.ComposeMessageActivity***".

6.1.5 RESUMEN DE RESULTADOS OBTENIDOS

A continuación se va a resumir el contenido de esta sección, en la que hemos investigado y analizado en profundidad el software con el que queremos interactuar. Todos los parámetros recogidos están contenidos en la tabla 112.

	Compresión	Escalado de imágenes	Llamada de comunicación
WhatsApp	Por defecto **80**.	Imágenes mayores de **1000** píxeles en una de sus componentes.	*com.whatsapp.ContactPicker*
Line	Por defecto **90**.	Imágenes mayores de **1000** píxeles en una de sus componentes.	*jp.naver.line.android.activity.selectchat.SelectChatActivity*
Spotbros	Entre **90** y **100** (usaremos **90**).	Imágenes mayores de **1000** píxeles en una de sus componentes.	*com.spotbros.activities.MainActivity*
MMS	Entre **95** y **50** (peso > **300KB**).	Imágenes mayores de **480x640** o tamaño superior a **300KB**.	*com.android.mms.ui.ComposeMessageActivity*
Email	Ninguno.	Ninguno.	A través del menú "**Enviar a**".

Tabla 112: Resumen atributos de las aplicaciones de mensajería.

Estos datos serán empleados en la aplicación de esteganografía objeto de este proyecto, sin embargo podrían ser empleados para el estudio o realización de otros trabajos similares.

6.2 PROCEDIMIENTO DE CIFRADO

Esta sección es una ampliación de la 5.3.3, en la que se describía a nivel de diseño cómo se iba realizar el cifrado del mensaje secreto. Aquí vamos a describir brevemente los algoritmos (en pseudo-código) que componen este proceso de seguridad.

Como se mostró en la figura 14, el proceso de seguridad comienza con el cifrado del mensaje, continúa con el desordenamiento del mismo y concluye con la codificación en *base64.*

6.2.1 MÉTODO DE CIFRADO

El método de cifrado principal recibe 2 parámetros principales: el mensaje a ocultar en forma de una estructura de bytes, y la contraseña como cadena de texto. Cuando se ha completado el proceso se devuelve otra estructura de bytes que representa el mensaje cifrado.

A continuación se muestra en muy alto nivel el procedimiento empleado:

```
// Método para cifrar un mensaje
Cifrar (mensaje, contraseña):
    ▪ Generación del "Salt":
        ○ Se concatena un Salt predeterminado con la contraseña.
        ○ Se genera el hash de dicha cadena utilizando la función resumen SHA-1.
        ○ Repetir proceso tantas veces como longitud tenga la contraseña:
            ▪ Se concatena el Salt por defecto al hash generado.
            ▪ Se genera el hash de la concatenación anterior usando SHA-1.
    ▪ Derivación de la contraseña:
        ○ Se emplea el algoritmo de derivación PBKDF2WithHmacSHA1.
        ○ Se deriva la contraseña 1000 veces utilizando el Salt generado.
        ○ La clave devuelta tiene 256bits de longitud.
    ▪ Cifrado del mensaje:
        ○ Se emplea el algoritmo de cifrado AES.
        ○ Se cifra el mensaje utilizando la clave derivada de 256 bits.
Se devuelve el texto cifrado.
```

Tabla 113: Pseudo-código del algoritmo de cifrado.

En el código anterior se muestra el método de cifrado que emplea la aplicación, este método ya había sido representado en la figura 15 del capítulo de diseño. Las partes más relevantes del cifrado son:

GENERACIÓN DEL *SALT*

Un ***salt*** es un dato aleatorio que suele emplearse en criptografía como entrada adicional para procesos relacionados con las funciones resumen o *hashes.* Suele emplearse también en entornos donde es necesario almacenar una contraseña en una base de datos, de manera que aunque un atacante obtuviera los datos de la misma no le sería sencillo invertir la función resumen, ya que se ha empleado un parámetro externo (*salt*) para aumentar la dificultad para *crackear* la información almacenadas.

En nuestro caso, el valor del *salt* es empleado como parámetro adicional de entrada para la función de derivación de la clave. Inicialmente se parte de un valor predeterminado en el código, que se concatena con la contraseña del usuario para posteriormente ser transformada

empleando el algoritmo de función resumen SHA-1. Se *hashea* esta cadena tantas veces como longitud tenga la contraseña del usuario.

Se ha decidido emplear un parámetro externo como es el *salt* para aumentar la complejidad del proceso de seguridad y hacer más difícil la labor de un atacante en caso de que obtuviera el criptograma.

DERIVACIÓN DE LA CONTRASEÑA

La derivación de contraseña tiene 2 objetivos bien definidos:

1. Producir una clave pseudo-aleatoria con una longitud determinada a partir de la contraseña del usuario, lo cual servirá de entrada para un proceso posterior de cifrado.
2. Incrementar la dificultad para romper un criptograma, ya que añade una carga computacional adicional (a través del número de iteraciones) que reduce la efectividad de un ataque de fuerza bruta ("*bruteforce*") contra la contraseña.

Se ha empleado la función de derivación "***PBKDF2WithHmacSHA1***", que recibe 4 parámetros para generar la clave:

- Contraseña del usuario.
- *Salt* obtenido de la función de generación explicada en el punto anterior.
- Número de iteraciones para la derivación que debe realizar.
- Longitud de la clave obtenida.

Se ha escogido 1000 como número de iteraciones porque:

- Genera claves suficientemente aleatorias y seguras.
- Requiere un tiempo de computación moderado para un dispositivo móvil.

Es importante velar por la velocidad y rendimiento de la aplicación además de la seguridad, ya que un número más elevado de iteraciones podría producir tiempos de procesamiento demasiado elevados teniendo en cuenta que el procesador donde se ejecuta la aplicación no tiene la capacidad de cómputo de un ordenador real.

Se ha realizado una medición sobre el tiempo de procesamiento que requiere la función de derivación en un dispositivo móvil con 1GHz de procesador, y los resultados arrojan que para un *salt* de 20 bytes y 1000 iteraciones se requiere un tiempo de procesamiento medio de 0.35 segundos.

CIFRADO AES

El algoritmo **AES** (*Advanced Encryption Standard*) es un sistema de cifrado simétrico por bloques pensado para las transmisiones seguras de mensajes a través de la red.

Este algoritmo se caracteriza por su velocidad, además de emplear muy poca memoria para realizar los cálculos. AES usa una red de sustitución-permutación para realizar el cifrado, sus bloques tienen un tamaño fijo de 128 bits, y las claves pueden ser de 128, 192 y 256 bits.

AES está considerado por la NSA (Agencia de Seguridad Nacional de los EEUU) como un algoritmo seguro para proteger información clasificada de nivel **secreto** si se usan claves de 128 o 256 bits. Actualmente no se considera roto dicho algoritmo pues no existe ningún ataque más rápido que fuerza bruta [25]. Para atacar al algoritmo AES con clave de 256 bits serían necesarias 2^{254} operaciones, lo cual es inviable con los recursos actuales.

En el módulo de cifrado que se ha implementado se ha empleado el algoritmo AES con una clave de cifrado de 256 bits. Internamente el procedimiento de codificación conlleva 14 rondas de procesamiento, por la longitud de la clave.

El uso de este algoritmo culmina un proceso de cifrado que se puede considerar como altamente seguro. El eslabón más débil de esta cadena de seguridad es el usuario, que al fin y al cabo es el que escoge la contraseña, que puede tener cualquier longitud en el caso de esta aplicación por cuestiones de usabilidad.

6.2.2 MÉTODO DE DESORDENAMIENTO

El algoritmo de desordenamiento empleado en el sistema es muy sencillo, a continuación en la tabla 114 se implementa su pseudo-código:

```
// Método para desordenar el mensaje cifrado
Desordenar (mensaje):
    ▪ Se copia el mensaje en una variable auxiliar 'Copia'.
    ▪ Se almacena en una variable 'Final' la longitud del mensaje.
    ▪ Se inicializa a 0 una variable 'Inicial' y otra 'contador'.
    ▪ Se repite el proceso tantas veces como longitud tenga el mensaje:
        ○ En la posición 'contador' del mensaje se almacena el carácter de
          'Copia' situado en la posición 'Contador'.
        ○ En la posición siguiente del mensaje se almacena el carácter de
          'Copia' situado en la posición 'Final'.
        ○ Se incrementan en una unidad las variables 'Inicial' y 'Final'.
        ○ Se incrementa en dos unidades la variable 'Contador'.
    ▪ Si la longitud del mensaje es impar, el último carácter del mensaje
      corresponde con el situado en la posición 'Final' de 'Copia'.
Se devuelve el mensaje desordenado.
```

Tabla 114: Pseudo-código del algoritmo de desordenamiento.

El algoritmo anterior va cogiendo progresivamente un carácter del principio del mensaje y después uno del final, de modo que el desordenamiento finaliza cuando se llega al carácter situado en la mitad del mensaje.

Se trata de un método de sustitución de los caracteres de un mensaje, de manera que éste queda ilegible. Es una 'medida de seguridad' adicional para aumentar la dificultad de ataque al criptograma. El método de ordenamiento funciona de la misma forma invirtiendo el proceso anterior.

6.2.3 MÉTODO DE CODIFICACIÓN

Se ha implementado el sistema de codificación **Base64** para finalizar el proceso de seguridad del mensaje.

El cifrado del mensaje devuelve una estructura de bytes que no puede ser introducida sin más en la imagen, ya que en muchos casos lo componen caracteres no imprimibles y que no pueden ser representados. Aquí es donde entra en juego la codificación en base64, que codifica cada uno de esos caracteres usando 64 como base. El resultado es una cadena de texto imprimible en el rango de caracteres A-Z, a-z, y 0-9, que se corresponden con la mayoría de los caracteres ASCII. De esta forma el mensaje quedará representado por el código de caracteres ASCII, que tiene un tamaño de **7 bits** por carácter.

Para poder codificar en base64 el mensaje cifrado se ha utilizado una clase auxiliar de JAVA que permite la codificación y decodificación de mensajes en distintos formatos como estructuras de bytes o cadenas de texto. La clase de JAVA empleada es de público uso y distribución (Base64.java versión 2.2 de Mikael Grev).

6.3 PROCEDIMIENTO DE OCULTACIÓN

En la presente sección se describirá la implementación del proceso de ocultación del mensaje cifrado en la imagen seleccionada por el usuario. En el capítulo de diseño se describieron los pasos necesarios para el subsistema de ocultación, ahora se mostrará a alto nivel cómo funciona específicamente los mecanismos de:

- Dispersión.
- Ocultamiento sin compresión.
- Ocultamiento con compresión.
- Corrección de errores.

6.3.1 DISPERSIÓN

Las técnicas esteganográficas incluyen mecanismos sofisticados para repartir la información sobre los píxeles de una imagen. Esta difusión protege los datos y los hace menos susceptibles de poder ser descubiertos por humanos o computadores que busquen patrones estadísticos de detección de mensajes ocultos.

En algunos casos, la dispersión de la información incrementa la resistencia a ser destruido por un atacante o por un algoritmo de compresión. Los algoritmos de dispersión a menudo distribuyen la información de manera que no son necesarios todos los bits para ensamblar el mensaje original, ya que introducen bits redundantes por si alguno queda dañado. En la sección de recuperación de errores comentaremos cómo crear sistemas de redundancia a nivel de bit para recuperarse de estos errores, sin embargo el objetivo de esta sección es definir el algoritmo de dispersión que mejor se adapte al proyecto.

En la imagen tenemos que ocultar 2 elementos:

- **La longitud del mensaje cifrado.**
- **El mensaje cifrado.**

Se ha decidido introducir la longitud del mensaje en la imagen por los siguientes motivos:

- Las marcas de fin de mensaje no funcionan muy bien, en el caso de que quedaran dañadas se buscarían dichas marcas por toda la imagen indefinidamente.
- Se puede identificar si un mensaje está dañado si la longitud extraída en el lado del receptor fuera superior a la permitida. Del mismo modo esto sirve para identificar si la contraseña introducida es errónea.
- Se optimizan mejor los recursos computacionales si se sabe desde el principio la longitud del mensaje que hay que recuperar.

Queda visto que la inclusión de la longitud dentro de la imagen tiene por objetivo facilitar el proceso de extracción en el extremo del receptor.

Antes de ocultar la información necesitamos obtener un mapa de localizaciones donde se especifique en qué posiciones de la imagen debe introducirse la información, lo llamaremos **mapa de dispersión**. Dicha estructura no puede tener los mismos valores siempre, pues si así

fuera los mensajes ocultos se ocultarían una tras otra vez en los mismos píxeles y por tanto sería trivial recuperar su contenido por parte de un atacante. Por ello se va a generar un mapa de dispersión pseudo-aleatorio de posiciones a partir de la contraseña introducida.

Una vez más la seguridad del sistema depende de la contraseña que escoja el usuario. En el pseudo-código de la tabla 115 se resume el funcionamiento básico del método de dispersión, cuyas entradas son:

- **La longitud de la imagen**, que indica el número de posiciones disponibles para almacenar información.
- **El número de elementos a ocultar**, que indica el número de posiciones que van a ser necesarias para albergar la longitud del mensaje, el mensaje en sí mismo y la redundancia necesaria.
- **La contraseña** para calcular el mapa de dispersión de manera pseudo-aleatoria.

El método de dispersión devuelve, cuando finaliza, una estructura que contiene tantas posiciones como elementos haya que ocultar. En cada una de estas posiciones se almacenará un entero que representa el byte de la imagen en el que se debe almacenar el dato que corresponda.

```
// Método de dispersión
Dispersión (Longitud_Imagen, Elementos_A_Ocultar, Contraseña):
    ▪ Se declara una estructura 'Posiciones' que almacenará los lugares donde se
      ocultarán los datos.
    ▪ Se declara una variable 'pass' que contiene el hash SHA-1 de la contraseña.
    ▪ Se inicializa una variable 'contador' a 0.
    ▪ Se repite el proceso tantas veces como 'Elementos_A_Ocultar' indique:
        o Se hashea 'pass' con SHA-1 y se almacena en 'pass'.
        o Se genera un número aleatorio 'random' utilizando el contenido de la
          posición 'contador' de 'pass'.
        o Mientras que la posición 'random' esté ocupada:
            - Se genera de nuevo un número aleatorio a partir de 'random'.
        o Se almacena en la posición 'contador' de 'Posiciones' el valor
          aleatorio contenido en 'random'.
        o Se marca esa posición de la imagen como ocupada.
        o Se incrementa 'contador' en una unidad.
Se devuelve 'Posiciones'.
```

Tabla 115: Pseudo-código del método de dispersión.

En el código anterior se observa cómo se rellena pseudo-aleatoriamente el mapa de dispersión utilizando como semilla la contraseña, que se *rehashea* tras obtener una nueva posición. Cada posición (byte) de la imagen sólo puede ser empleada para almacenar un dato, que por lo general será un bit. De modo que se guarda en una variable global las posiciones de la imagen que ya han sido asignadas. Por ello, en el momento en que se genera pseudo-aleatoriamente una posible posición en el mapa de dispersión debe comprobarse primero si dicha posición había sido previamente asignada. De ser así, se produce un **rebote**, y se genera un nuevo número a partir del anterior tantas veces como sea oportuno hasta encontrar una posición libre.

Se ha establecido que tras cada vuelta se regenere una función resumen nueva partiendo de la anterior para añadir la mayor **aleatoriedad** posible al proceso de selección de posiciones. De hecho se ha comprobado que de esta forma se obtienen menos rebotes que utilizando otros métodos.

Es importante destacar que en el caso de que se genere el mapa de posiciones para un modo que suponga compresión (como es WhatsApp, Line y Spotbros), será fundamental obtener posiciones múltiplo de 3. Esto es porque la ocultación con compresión necesita posicionarse en el primer byte de cada píxel. Para los modos que no requieren compresión se puede seleccionar cualquier byte de la imagen.

El mapa de dispersión generado en este paso será empleado por el método de ocultación para introducir en las posiciones establecidas los elementos necesarios (longitud y mensaje cifrado con redundancia).

6.3.2 SISTEMA PARA ENVÍO SIN COMPRESIÓN DE IMAGEN

TÉCNICA EMPLEADA

En el ámbito de la esteganografía, la técnica de ocultación más famosa es la **LSB** o de bit menos significativo. Es una técnica de sustitución que trabaja sobre el dominio espacial, es decir, directamente sobre la imagen. Se trata de introducir la información que deseemos ocultar en los bits menos significativos del medio portador, de esta manera la repercusión visual será mínima. Sin embargo, la información introducida utilizando la técnica LSB es altamente vulnerable y podría ser destruida si se aplica una ligera modificación en la imagen portadora, como sucede con la compresión JPEG.

Para mostrar el funcionamiento de la técnica se muestra a continuación en la figura 46 cómo se oculta el carácter 'A' en los bits menos significativos de una imagen. El carácter necesita 8 bits así que modifica el valor del último bit de cada byte en los casos en los que no coincida (cuadrados rojos).

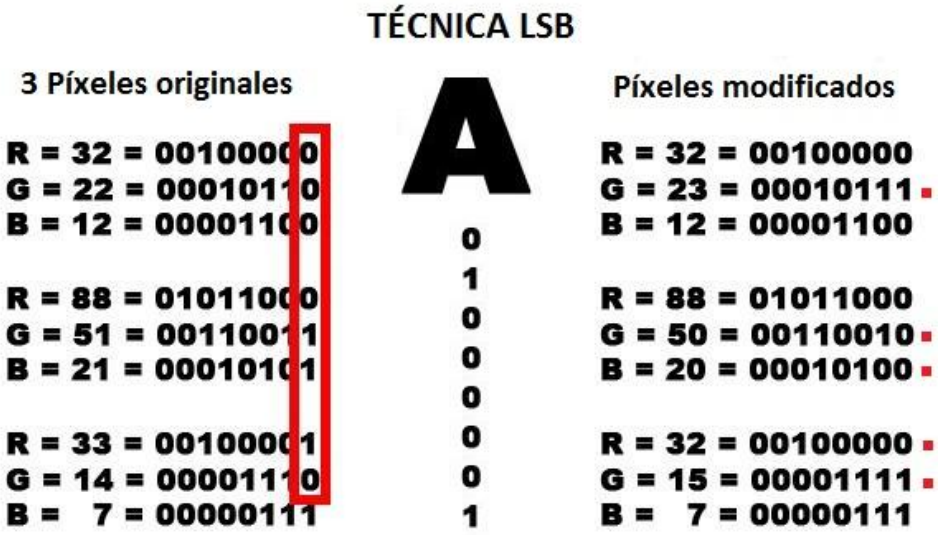


Ilustración 46: Técnica LSB.

Esta técnica es la que se ha empleado en nuestro sistema para ocultar la información que no va a ser comprimida posteriormente, como es el caso del correo electrónico y del MMS (una vez se ha reducido su tamaño a menos de 300KB).

PROCEDIMIENTO

El método de ocultamiento recibe 3 parámetros:

- **El elemento a ocultar**, que será primero la longitud del mensaje y después él mismo.
- **La imagen**, se recibe como una estructura de bytes.
- **El mapa de dispersión**, que indica dónde se debe ocultar cada bit.

Para realizar la ocultación se siguen los siguientes pasos:

1. Se selecciona en el mapa de dispersión la posición indicada por el contador de posiciones, que inicialmente está a cero y se incrementa cada vez que se necesita ocultar un bit.
2. Se obtiene el byte de la imagen al que hace referencia dicha posición.
3. Se coge el byte que corresponda ocultar de nuestro mensaje, cada carácter ocupa un byte.
4. Se sustituye el bit que se necesita ocultar por el menos significativo del byte seleccionado previamente de la imagen.

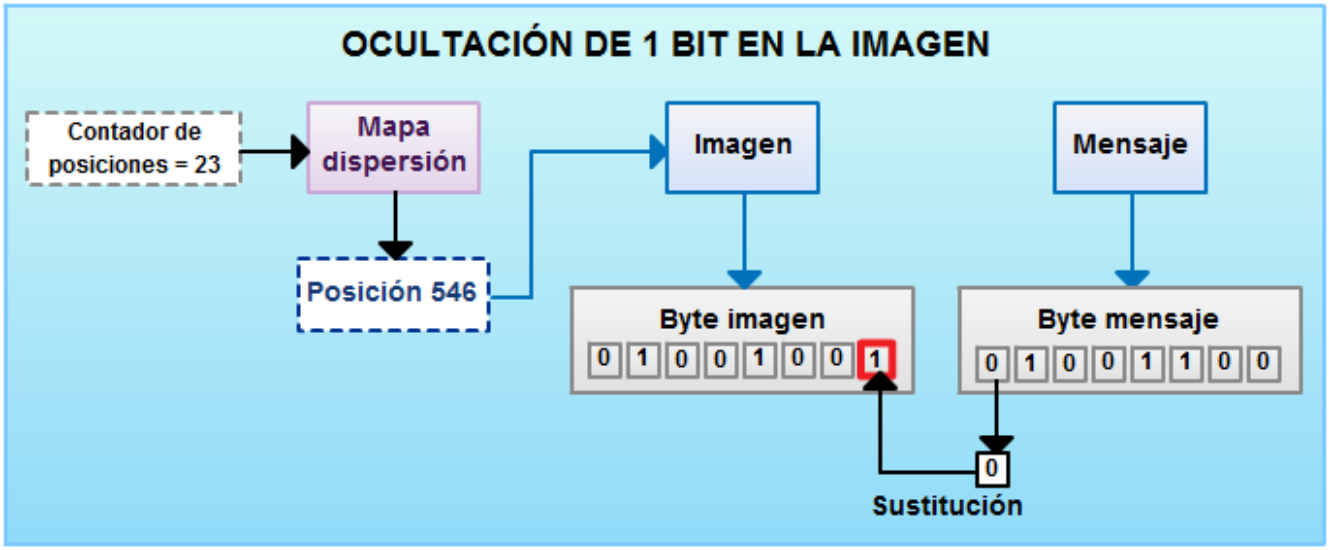


Ilustración 47: Ejemplo de ocultación de 1 bit en la imagen sin compresión.

En la figura 47 se ilustra un ejemplo del proceso de ocultación, en el cual ya se han ocultado 22 elementos anteriormente. El contador de posiciones indica al mapa de dispersión que el siguiente byte de la imagen en el que se va a ocultar un nuevo bit se encuentra en su elemento número 23. Una vez se resuelve ese valor, se recupera el byte de la imagen al que apunta dicho valor, 546 en este caso. La sustitución se realiza sobre el último bit del byte 546 (marcado en rojo), que originalmente era un 1 y ahora va a pasar a ser un 0.

Una vez que se ha introducido un bit de información en la imagen, es necesario introducir su redundancia por si se producen errores. El proceso de ocultación de redundancia funciona de manera idéntica que el anteriormente descrito pero con una diferencia: en lugar de ocultar un bit se ocultan 3, en las últimas posiciones del siguiente byte apuntado por el mapa de dispersión.

En la figura siguiente se ilustra el proceso de introducción de la redundancia:

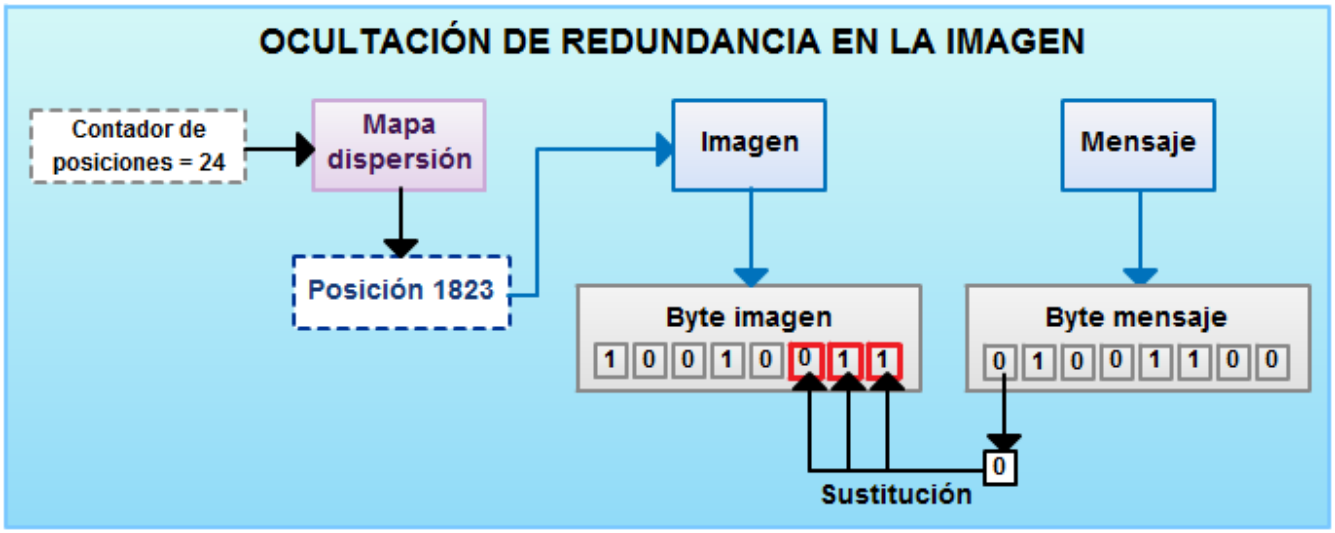


Ilustración 48: Ejemplo de ocultación de redundancia en la imagen sin compresión.

La ilustración 48 representa el paso posterior al proceso mostrado en la figura 47: tras haber introducido un bit del mensaje en la imagen, se incrementa el contador de posiciones, de modo que se selecciona el siguiente byte de la imagen donde se introducirá la redundancia (byte 1823), y se sustituyen en dicho byte las tres posiciones menos significativas por el bit que hemos ocultado antes, el cero. En este ejemplo se modificarían sólo los 2 últimos bits, que están inicialmente puestos a 1.

Cada bit de nuestro mensaje sustituirá 4 bits en la imagen, uno normal y tres de redundancia. La introducción de la redundancia supone un cambio de color de hasta 7 tonalidades como máximo en una de las componentes del píxel correspondiente. Teniendo en cuenta que cada componente RGB de un píxel puede adquirir 255 valores diferentes, no tiene una repercusión visual notoria esta inserción.

Una vez que se ha ocultado la longitud y el mensaje en la imagen finaliza el proceso de ocultación, tras el cual se realizan las siguientes acciones:

1. Se reconvierte la estructura formada por bytes de la imagen en un **mapa de bits**.
2. A dicho mapa de bits se le aplica una **compresión sin pérdidas** con el formato **PNG**.
3. La imagen se guarda en el directorio "**Estegano**" de la galería.
4. Se envía la imagen resultante por **MMS** o **email**.

6.3.3 SISTEMA PARA ENVÍO CON COMPRESIÓN DE IMAGEN

INTRODUCCIÓN

En esta sección se tratará de realizar un procedimiento similar al explicado para los sistemas sin compresión, con la única diferencia que las imágenes resultantes de este proceso tienen que estar preparadas para sufrir las compresiones que WhatsApp, Line y Spotbros van a imponer.

En primer lugar se propondrá una técnica esteganográfica efectiva para que las imágenes puedan soportar las compresiones impuestas por las aplicaciones de mensajería instantánea. Después se tratará de refinar dicha técnica para que los resultados obtenidos sean igual de efectivos y la repercusión visual resultante de aplicar dicha técnica no sea tan notoria como en el primer caso.

Finalmente se concluirá explicando el procedimiento final empleado en el sistema.

TÉCNICA EMPLEADA

Es bien sabido que las técnicas LSB, que trabajan con los bits menos significativos de una imagen, no pueden ser utilizadas en los entornos en los que se impone una compresión. Esto es debido a que el proceso de **compresión JPEG** realiza muchas modificaciones sobre la imagen, entre esas modificaciones se encuentra el cambio de modelo de color (RGB a YUV), la cuantificación de los coeficientes obtenidos tras aplicar la transformada del coseno (DCT) sobre bloques de 64 píxeles de la imagen, y la codificación de entropía que se produce.

Todo intento que se produzca para insertar información en los bits menos significativos de la imagen no servirán para nada. Es en este punto donde surge la idea de emplear los **bits más significativos** (**MSB**) de los píxeles para almacenar la información secreta. Se trata de que se pueda decidir si un píxel contiene un 0 o un 1 tras realizar una compresión sobre la imagen. Para ello se variará radicalmente el color de los píxeles: las 3 componentes que forman el color del píxel se fijarán en su valor máximo (255) para representar un 1, o en su valor mínimo (0) para representar un 0. De esta forma si se pretende introducir un bit con valor 1 en un píxel se tendrán que establecer los 3 bytes que lo colorean a un valor de 255, el resultado obtenido será un píxel blanco. En caso de que se pretenda introducir un bit a 0, se pondrán las componentes RGB del píxel a 0, y el píxel resultante será negro.

Utilizando esta técnica se puede decidir en el lado del receptor si cierto píxel es más blanco que negro o viceversa. Basta con comparar los valores de cada una de sus componentes con un umbral, que establece si dicha componente es clara u oscura. Se decidirá que un píxel es blanco cuando al menos dos de sus componentes sean determinadas como claras, y por tanto sus valores estarán más próximos a 255 que a 0.

No es una técnica muy elegante pues la repercusión visual en la imagen tras la inserción es muy evidente, se ven tantos píxeles blancos o negros como bits tiene la información ocultada.

En la siguiente figura se muestra un ejemplo del funcionamiento de este método:

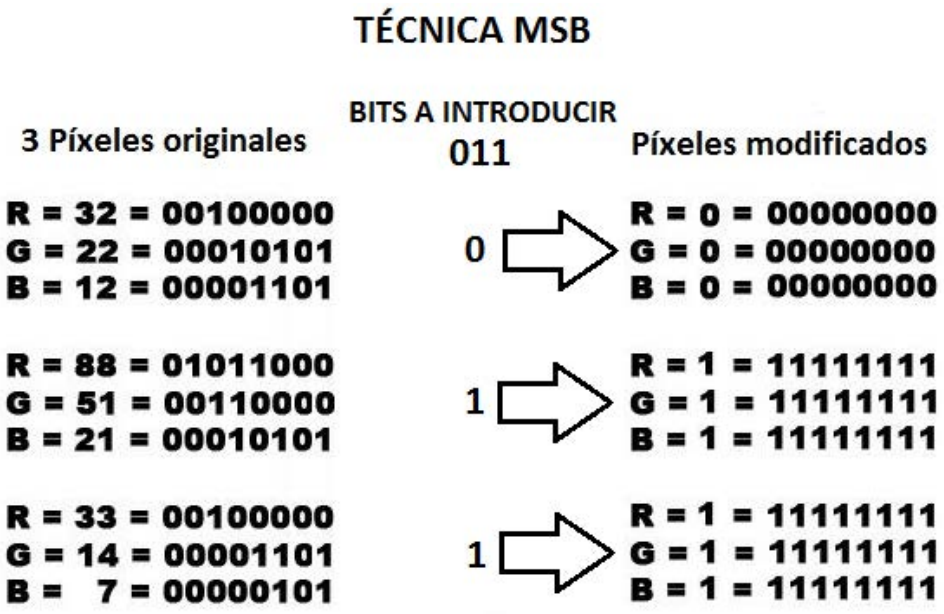


Ilustración 49: Ejemplo técnica MSB.

REFINAMIENTO DE TÉCNICA

La técnica anterior funciona correctamente, pues ha sido capaz de superar las pruebas de compresión realizadas con WhatsApp. Sin embargo, cualquiera que observe una imagen con tal variación de color entre píxeles aislados sospechará de la existencia de un posible mensaje.

Para hacer menos notoria la repercusión visual que genera la técnica MSB, se propone un método derivado que plantea modificar lo menos posible el color de los píxeles que sean empleados para almacenar el mensaje secreto. Se trata por tanto de aumentar o reducir la tonalidad de la componente roja, verde y azul de un píxel hasta que pueda ser considerada como un 1 o un 0 en el extremo receptor.

Pero, y aquí viene la pregunta fundamental: ¿Cómo decidir si un píxel va a ser interpretado como un 1 o como un 0 antes de enviar la imagen? La respuesta es sencilla: comprimiendo y descomprimiendo la imagen, para posteriormente analizar si el valor de sus componentes está más cercano a 255 o a 0. Esta compresión y descompresión se realiza obviamente en el lado del cliente durante la ejecución del programa. En el caso de que se obtenga un valor distinto al esperado se aumentarán o disminuirán de nuevo los valores de las componentes RGB que forman el píxel para posteriormente repetir el proceso. En el momento que se obtenga el valor esperado tras la compresión y descompresión de la imagen, el valor de ese píxel quedará fijo.

Se trata de un proceso bastante lento dependiendo de la medida en que se aumente o disminuya el valor de los bytes de un píxel. Esa medida recibirá el nombre de "**parámetro de modificación**", e indicará la cantidad que se debe añadir o sustraer a cada uno de los tres bytes de cada píxel en caso de que no superen la comprobación de ocultación descrita anteriormente.

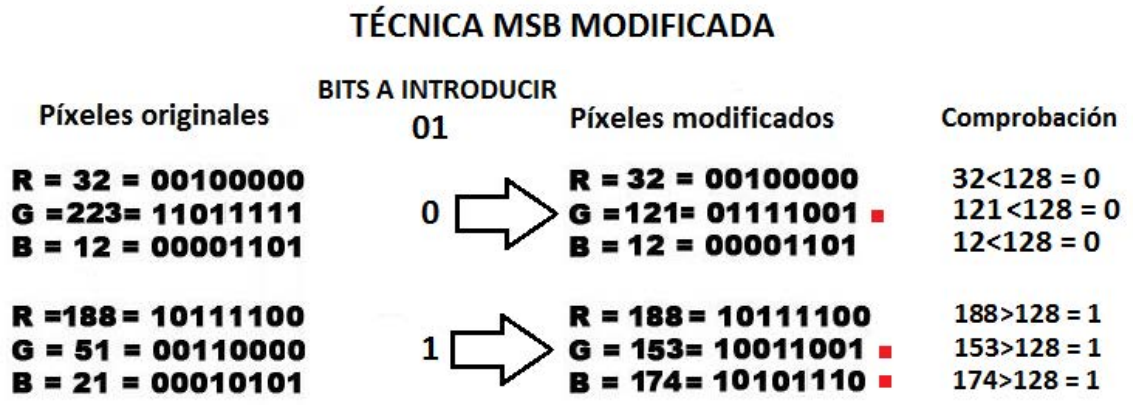


Ilustración 50: Ejemplo técnica MSB modificada.

La figura 50 muestra un ejemplo de la técnica presentada: se desea introducir un 0 en el primer píxel, para ello modifica sólo la componente cuyo valor es superior a 128 (la verde=223) y se le resta el parámetro de modificación en dos ocasiones (el parámetro decidido es 51) hasta pasar el umbral de 128, que lo define como un 0. En el segundo píxel ocurre el caso contrario, se desea introducir un 1, y se aumenta en dos ocasiones la componente verde y en tres la componente azul, hasta que superan el umbral de 128 que representan un 1. Las componentes de los píxeles que tenían el valor requerido desde el principio no se han modificado, y conservarán su tonalidad.

Esta técnica permite que muchos de los píxeles que se necesiten modificar para adjuntar el mensaje oculto conserven un color mucho más aproximado al que tenían originalmente que en el caso de convertirlos a blanco o negro.

A pesar de ello, la técnica presenta 2 problemas importantes en este momento:

- Se pueden producir **errores** en el envío que cambien levemente la tonalidad establecida de los píxeles portadores, de manera que el receptor es incapaz de recuperar el mensaje.
- En ocasiones, cuando las componentes de un píxel tienen valores muy cercanos a 128 cualquier mínima modificación producida por el influjo de los píxeles adyacentes puede aumentar o disminuir su magnitud lo justo para que el bit se pierda. Como hemos visto, durante el proceso de compresión JPEG se divide la imagen en bloques de 64 píxeles, de manera que si los píxeles de cierto bloque tienen unas características muy distintas respecto a un píxel modificado por nuestro sistema que se encuentre en dicho bloque, es posible que se **altere el color** del píxel y pase a ser más oscuro o más claro que anteriormente y se pierda el bit oculto.

El primero de los problemas será resuelto introduciendo redundancia adicional para cada bit, como se comentará posteriormente.

El segundo problema supone que tendremos que:

- Ampliar la diferencia a la hora de determinar si un píxel es más claro o más oscuro, es decir, establecer el umbral para los bits a 1 en valores superiores a **140**, y para los bits a 0 en valores inferiores a **115**.

- En caso de ser incapaces de introducir un mensaje sin que se produzcan errores se procederá a la **decoloración** de la imagen, tras la cual se reintentará el proceso de ocultación.

Siguiendo las soluciones que se acaban de proponer debería ser posible realizar el procedimiento que nos permitiese ocultar información en una imagen introduciendo la menor cantidad posible de repercusión visual o ruido.

PROCEDIMIENTO

La función de ocultación que se ha implementado recibe los siguientes parámetros:

- **El mensaje** que se pretende ocultar.
- **La imagen** en forma de estructura de bytes.
- **El mapa de dispersión** para localizar las posiciones donde se debe ocultar cada bit.

Antes de adentrarnos en el funcionamiento del método de ocultación se deben establecer los siguientes parámetros:

- El **parámetro de modificación** establecido será **51**, por lo que el número máximo de veces que puede ser incrementado o reducido el valor de una componente de un píxel es de **5** (5x51=255).
- El **factor de decoloración** de la imagen será **2**. Si no se ha podido introducir el mensaje secreto en la imagen, se reintentará de nuevo reduciendo la saturación de la imagen progresivamente. En el peor de los casos, tras 5 pasadas reduciendo el color de la imagen, el mensaje se adjuntará en una imagen en escala de grises (sin color).

El procedimiento funciona de la siguiente manera:

1. A partir de un byte del mensaje se generan 11 bits de redundancia.
2. Se almacenan los 11 bits de redundancia de todos los caracteres del mensaje.
3. Se introduce en los píxeles apuntados por el mapa de dispersión el valor correspondiente (para la primera vez no se hace nada, se dejan intactos los píxeles de la imagen).
4. Se comprime la imagen con el parámetro de compresión correspondiente (80 WhatsApp, 90 Line y 90 Spotbros).
5. Se descomprime la imagen y se comprueba si los píxeles tienen el valor esperado.
6. Si tienen el valor esperado se avanza al paso 11. Si no es así se continúa.
7. Si se ha reintentado el proceso 5 veces consecutivas ir al paso 10. Si no se continúa.
8. Se comprueba qué píxeles no tienen el valor esperado.
9. Para esos píxeles se aumenta o reduce la magnitud de sus componentes RGB en 51 unidades (no pueden superar los límites de 0 o 255). Y se vuelve al paso 4.
10. Si el factor de coloración es 0 el sistema ha sido incapaz de ocultar la información y se notifica al usuario. Si es mayor que 0 se reduce en dos unidades y se vuelve al paso 3.
11. Si se ha logrado ocultar todos los bits en los píxeles sin ningún error se comprime la imagen original (con los valores de los píxeles que han funcionado) en formato PNG.
12. Se guarda la imagen en la carpeta “Estegano” de la galería.
13. Se envía a través de WhatsApp, Line o Spotbros según proceda.

En el siguiente diagrama de flujo se muestran los pasos descritos anteriormente:

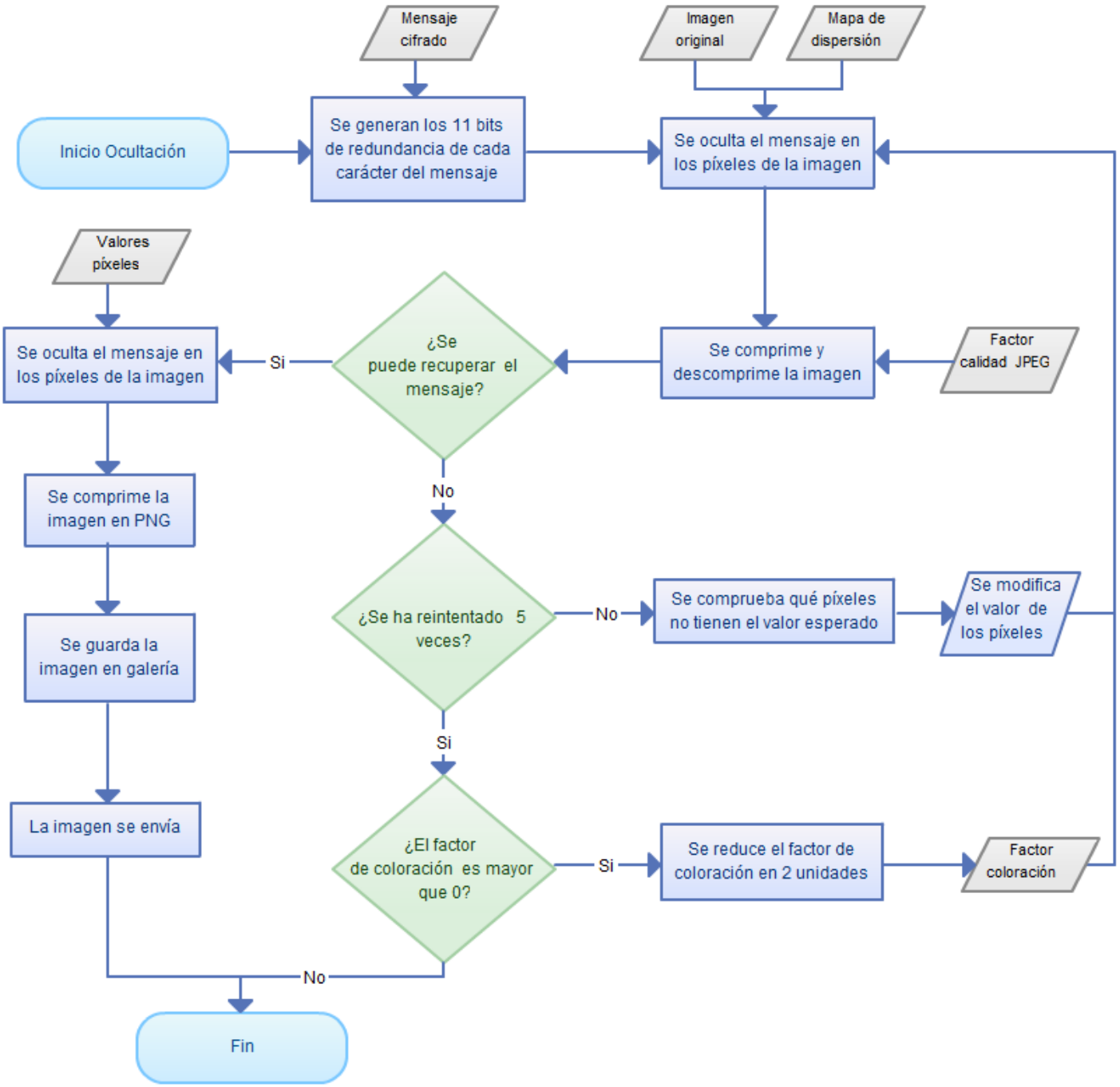


Ilustración 51: Diagrama de flujo para el método de ocultación de mensajes en imágenes con compresión.

La primera vez que se ejecuta el método de ocultación se realiza el siguiente procedimiento: Se comprime y descomprime la imagen original, se comprueba qué píxeles de la imagen, en los que se tiene que ocultar el mensaje, portan por defecto el valor que tiene que ser ocultado. De manera que los píxeles que cumplan con esa condición no necesitarán ser modificados en los siguientes pasos. Los que no posean el valor que indica el mensaje se modificarán.

La modificación de los píxeles consiste en incrementar o reducir el valor de sus componentes RGB en 51 unidades (dependiendo si se necesita ocultar un 1 o un 0 en dicho píxel). Se introducirán los nuevos valores modificados en los píxeles correspondientes para repetir el

proceso de compresión y descompresión, tras el cual aquellos que no tengan el valor adecuado se volverán a modificar. En el peor de los casos, tras 5 rondas los píxeles erróneos tendrán un valor de 255 o 0 en todas sus componentes (serán negros o blancos).

Si aún así es imposible recuperar el valor deseado se procederá con la decoloración de la imagen. Esta decoloración consiste en reducir la saturación de la imagen, es decir, su color. De este modo los píxeles que tenían los colores más vivos se tornarán más grises, y será más probable que se pueda introducir el mensaje en la imagen sin errores.

Cuando se ha decolorado la imagen hasta el valor 0 (escala de grises), si no ha sido posible insertar el mensaje se mostrará un mensaje al usuario indicándole lo sucedido. En la mayoría de casos debería poderse ocultar el mensaje sin necesidad de decolorar la imagen, pero en los casos en los que posee colores muy vivos o el mensaje a introducir es muy largo es posible que se tenga que realizar este procedimiento.

Como resultado final obtenemos una imagen en la que sólo unos pocos píxeles son visualmente distintos de sus adyacentes, ya que la mayoría de los se han modificado tienen un color muy similar al que poseían originalmente.

Se ha escogido el valor 51 como parámetro de aumento o disminución de los píxeles ya que es múltiplo de 255 y sólo supondría 5 iteraciones que un píxel modificara su valor completamente, lo que lo hace más rápido que si se escogiera un parámetro más pequeño. Se ha intentado mantener un equilibrio entre la velocidad de ocultación y la imperceptibilidad del ruido introducido en la imagen.

Es importante destacar que este procedimiento emplea 11 píxeles de la imagen para introducir un solo carácter, es decir, que si el mensaje cifrado tiene 10 caracteres se ocuparán 110 píxeles para realizar la ocultación. Cuanta más información sea introducida mayor será la repercusión visual resultante.

Pruebas realizadas indican que este procedimiento mejora de manera sobresaliente los resultados obtenidos con la técnica MSB original. Los píxeles que no necesitan modificar su valor son un **17% de media**, la mayoría de los píxeles (35%) requieren 102 desplazamientos, y son sólo una minoría (0.1%) los que de media necesitan 255 desplazamientos (que equivale de pasar del blanco al negro o viceversa).

EXTRACCIÓN DEL MENSAJE

En el lado receptor, para poder recuperar el mensaje se realizan los siguientes pasos:

1. Se obtiene el valor de los píxeles apuntados por el mapa de dispersión.
2. Se decide que un píxel contiene un 1 si el valor de al menos dos de sus componentes es superior a **130 unidades**. Si es inferior a ese valor se considera que el bit oculto es un 0.
3. Una vez se han recogido los 11 bits que componen un carácter se comprueba si la redundancia es correcta, si no es así se corrigen los errores.
4. Cuando se han recuperado todos los caracteres finaliza la extracción.

6.4 PROCEDIMIENTO DE RECUPERACIÓN DE ERRORES

En el sistema de información implementado ha sido necesario emplear **códigos de corrección de errores**. Estos códigos son útiles para tratar los errores introducidos por un posible atacante, un canal de transmisión ruidoso, o por la compresión de la imagen. A pesar de ello, no existe una medida que prediga el daño que recibirá nuestro fichero digital portador.

Se han utilizado dos métodos distintos para introducir códigos de corrección de errores, a continuación se explican:

6.4.1 CORRECCIÓN DE ERRORES PARA ENVÍO SIN COMPRESIÓN DE IMAGEN

En el caso de las imágenes que no van a sufrir compresión JPEG, tras su generación se va a emplear un mecanismo muy sencillo para corregir los errores que puedan producirse.

La información se oculta en los bits menos significativos, con lo que existe un riesgo mayor de perder el mensaje que si se emplearan los bits más significativos. Por cada bit ocultado se introducen otros 3 bits de redundancia en el siguiente píxel que marque el mapa de dispersión, el procedimiento que se emplea en el extremo receptor para detectar y corregir errores es el siguiente:

1. Se extraen el bit oculto y sus 3 bits de redundancia.
2. Se toma como **bit de comprobación** aquel que se encuentre en al menos 2 de los 3 bits de redundancia.
3. Se compara este valor de redundancia con el bit oculto extraído, si son iguales se confirma que no se ha producido ningún error. Si son distintos se toma el valor de la **redundancia** y se considera que se ha **corregido un error**.

Se trata de un procedimiento sencillo pero efectivo. Se ha comprobado en envíos por email y MMS que en mensajes con muchos caracteres de longitud se pueden llegar a corregir una gran cantidad de errores utilizando esta técnica.

6.4.2 CORRECCIÓN DE ERRORES PARA ENVÍO CON COMPRESIÓN DE IMAGEN

En el caso de que se necesite enviar las imágenes a través de WhatsApp, Line o Spotbros es necesario utilizar una técnica de corrección de errores más sofisticada. Se van a emplear **códigos *Hamming*** para la creación de códigos que corrijan y detecten errores, ya que son: fáciles de implementar, efectivos y elegantes.

El código *Hamming* más empleado es el llamado ***Hamming*(7,4)**, en el que se agregan tres bits adicionales de comprobación por cada cuatro bits de datos del mensaje. Este código es capaz de corregir cualquier error de un solo bit, pero si hay más de un bit erróneo, se corrige de manera incorrecta.

En nuestro caso, necesitamos generar un código de corrección de errores para los 7 bits que representan cada carácter codificado en ASCII, para ello se emplearán códigos ***Hamming*(11,7)**, que introducen 4 bits adicionales de paridad para detectar y corregir cualquier error que se

produzca en los otros 7 bits que transmitimos. A continuación se muestra un ejemplo en el que se crea un código de recuperación para los 7 bits "0110101", se ha utilizado la letra "d" para representar los bits de datos, y la letra "p" para los bits de paridad:

	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7
Bits sin paridad:			**0**		**1**	**1**	**0**		**1**	**0**	**1**
p_1	**1**		0		1		0		1		1
p_2		**0**	0			1	0			0	1
p_3				**0**	1	1	0				
p_4								**0**	1	0	1
Bits con paridad:	**1**	**0**	0	**0**	1	1	0	**0**	1	0	1

Tabla 116: Ejemplo de codificación *Hamming* (11,7) [26].

El código de recuperación obtenido es "10001100101", donde el primer, segundo, cuarto y octavo bit representan los bits de paridad introducidos. Ahora imaginemos que el código anterior se corrompe y el bit de la posición 11 pasa de ser un 1 a un 0, se realizaría el procedimiento indicado en la tabla 117 para la detección del error.

	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	Paridad	Bit paridad
Cadena recibida:	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	
p_1	**1**		0		1		0		1		0	**Fallo**	**1**
p_2		**0**	0			1	0			0	0	**Fallo**	**1**
p_3				**0**	1	1	0					Correcto	0
p_4								**0**	1	0	0	**Fallo**	**1**

Tabla 117: Ejemplo de corrección de errores con *Hamming*(11,7).

Una vez se ha detectado un error de paridad se realiza el siguiente procedimiento para corregir el error:

1. Se toman los bits de paridad recibidos: 1, 1, 0, 1 (columna "Bit paridad" de la tabla 117).
2. Se ordenan (el primer bit de paridad es el situado más a la derecha, y el último es el situada más a la izquierda): 1, 0, 1, 1.
3. Se calcula el valor numérico que representa dicha cadena binaria: $2^3 + 2^1 + 2 = \mathbf{11}$.
4. El valor obtenido representa el bit erróneo recibido, en nuestro caso el bit 11.
5. Se invierte su valor en la cadena recibida: si estaba a 0 se pone a 1.
6. Se eliminan los bits de paridad para obtener la palabra original: 0110101.

En el caso de que se recibiera un bit de paridad erróneo su detección sería más sencilla, ya que todas las comprobaciones de paridad serán correctas menos la que posea el bit alterado.

Este procedimiento se emplea en el lado del emisor para detectar si la información que se va a enviar tiene algún error, y en el lado del receptor para corregir todos los bits erróneos. El método de ocultación presentado debe la efectividad, en gran parte, al mecanismo de generación, detección y corrección de códigos de error presentado.

7 EVALUACIÓN Y RESULTADOS

En este capítulo se presentan las evaluaciones y resultados de la aplicación presentada. Los objetivos de este capítulo son:

- Evaluar las propiedades esteganográficas de la aplicación.
- Realizar un estegoanálisis a las imágenes generadas.
- Evaluar el rendimiento de la aplicación.
- Evaluar la seguridad de la aplicación.
- Estudiar el impacto de la publicación de la aplicación en *Google Play*.

En las siguientes secciones se tratarán los objetivos marcados haciendo especial hincapié sobre los algoritmos esteganográficos propuestos.

7.1 ESTEGANOGRAFÍA

La calidad de un sistema esteganográfico se evalúa por sus 3 características esenciales:

- **Capacidad:** La cantidad de información que es capaz de ocultar.
- **Indetectabilidad:** La dificultad para detectar la presencia de información.
- **Robustidad:** La capacidad para resistir ataques y corregir errores.

En la aplicación desarrollada se ha intentado mantener un equilibrio entre las tres propiedades anteriormente citadas y que pueden observarse en la figura 52.

Ilustración 52: Propiedades de una aplicación esteganográfica.

En los siguientes apartados se van a evaluar individualmente cada una de estas propiedades para los 2 algoritmos de ocultación propuestos, así se comprobará cuáles son los puntos fuertes de cada uno y qué debe mejorarse.

7.1.1 OCULTACIÓN SIN COMPRESIÓN DE IMAGEN

En primer lugar se evaluarán las propiedades anteriormente descritas para el subsistema de ocultación que no requiere compresión, éste era el que nos permitía los envíos por correo electrónico y mensajes multimedia.

CAPACIDAD

Muchos sistemas esteganográficos buscan poder ocultar la mayor cantidad de información posible en la imagen portadora, el mayor inconveniente que esto tiene es que cuanta más

información se oculte mayor será la deformación final de la imagen, y por tanto más evidente es la existencia de datos en su interior.

En la aplicación implementada se ha limitado en 1000 caracteres la capacidad de la imagen portadora. Se decidió desde un principio que esa cantidad de información era suficiente para enviar mensajes secretos entre dispositivos móviles. Además, hay que tener en cuenta que sólo se oculta texto, por lo que no es necesario extender en principio ese límite.

Para evaluar la capacidad de ocultamiento del algoritmo se empleará la medida de **bits por píxel (bpp)**. Esta medida relaciona la longitud del mensaje oculto con el número de píxeles de la imagen. Los bits por píxel indican la cantidad de bits de información que hay ocultados por cada píxel de la imagen.

Para hallar la cantidad de bits de información que se pueden ocultar en un píxel se deben conocer primero qué imagen se toma como referencia, en este caso se empleará una imagen estándar de 800 píxeles de ancho por 600 píxeles de alto. La imagen posee 480000 píxeles para albergar información en su interior.

El algoritmo de ocultación sin compresión utiliza:

- **32 Bits para ocultar la longitud** (un número entero).

  A estos 32 bits de longitud hay que sumarle la cantidad de redundancia que se adjunta. Recordemos que por cada bit ocultado en este modo se introducían 3 bits de redundancia adicionales en otro píxel. Por tanto estamos hablando de $3 \times 32 = 96$ bits de redundancia. Se requieren por tanto 128 bits para introducir la longitud en el mensaje, que se distribuyen en 64 píxeles distintos.

- **8 Bits para ocultar cada carácter cifrado**.

  Al igual que ocurre con la longitud, también es necesario incluir los bits de redundancia que se emplean como códigos de corrección para el mensaje. Se obtiene que el mensaje ocupa un total de $L \times 4 \times 8$ bits en la imagen, donde L es la longitud del mensaje, 4 la cantidad de bits que se introducen por cada carácter (1 normal y 3 de redundancia), y 8 el número de bits que ocupa un carácter.

  Se tomarán 5 valores como ejemplo para la longitud del mensaje, de manera que podamos obtener la cantidad de bits por píxel para cada caso, todo ello sobre una imagen de 800x600 píxeles:

Longitud del mensaje	Bits del mensaje	Bits totales con longitud	Bits por píxel (bpp)
20 caracteres	640	768	**0.0016**
50 caracteres	1600	1728	**0.0036**
100 caracteres	3200	3328	**0.0069**
500 caracteres	16000	16128	**0.0336**
1000 caracteres	32000	32128	**0.0669**

Tabla 118: Bits por píxel para el algoritmo de ocultación sin compresión.

Se observa que en el peor de los casos, cuando se ocultan 1000 caracteres de información la cantidad de bits por píxeles es baja. Esto tiene las siguientes lecturas:

- La dispersión de la información por la imagen es mayor que si se ocultaran más bits.
- La percepción de que hay un mensaje oculto es inexistente, pues sólo se emplean los bits menos significativos (hasta el tercer plano en el caso de la redundancia).
- Podría ampliarse la capacidad de almacenamiento.

Hay que decir que la prueba anterior se realizó con una imagen cuya resolución era muy alta (800x600 píxeles), puede darse el caso de que se empleen imágenes de menor tamaño. La cantidad de bits por píxel aumentará cuanto más pequeña sea la imagen en la que se vaya a realizar esteganografía.

Es necesario mencionar que toda inserción de información requiere un número mínimo de píxeles disponibles en la imagen, de manera que si no se satisface esa cantidad no podrá ocultarse el mensaje. En la siguiente tabla se muestra la cantidad de píxeles requeridos para insertar información:

Longitud del mensaje	Píxeles necesarios	Tamaño de imagen
20 caracteres	384	20x20 píxeles
50 caracteres	864	30x30 píxeles
100 caracteres	1664	41x41 píxeles
500 caracteres	8064	90x90 píxeles
1000 caracteres	16064	127x127 píxeles

Tabla 119: Píxeles requeridos para la inserción de distintas longitudes.

Como se ha podido observar es posible introducir una gran cantidad de información hasta en imágenes tan pequeñas como iconos.

Se queda abierta la posibilidad de poder introducir más información en futuras mejoras de la aplicación, ya que la longitud del mensaje es un entero de 4 bytes, lo que nos permitiría poder adjuntar mensajes de hasta 65535 caracteres, aunque para el proyecto actual sólo es posible insertar 1000 como máximo.

INDETECTABILIDAD

La indetectabilidad de un sistema esteganográfico consiste en el grado de perceptibilidad de un mensaje oculto. Esto incluye los ataques visuales de los humanos y los ataques estadísticos de los computadores.

Una imagen esteganografiada debe tener la menor distorsión posible respecto a la imagen original, es decir, que el ruido introducido durante la fase de ocultación no reduzca la calidad de la imagen en exceso. Para medir la distorsión de una imagen se emplea la "**relación señal a pico radio"** o **PSNR**.

El PSNR da una relación del grado de supresión de ruido que existe entre una imagen original y una procesada, proporcionando una medida de calidad. El PSNR se mide en decibelios (dB) y emplea el error cuadrático medio (MSE) para hallar su valor. Existe un mapa heurístico que

permite medir la calidad subjetiva de una imagen basándonos en el número de decibelios que represente el PSNR, se muestra en la tabla 120.

PSNR (dB)	Calidad subjetiva
>37	5 (Excelente)
31 - 37	4 (Bueno)
25 – 31	3 (Regular)
20 – 25	2 (Pobre)
<20	1 (Malo)

Tabla 120: Relación entre PSNR y calidad subjetiva.

La calidad subjetiva se mide en una escala del 1 al 5, siendo el 1 el factor de calidad más bajo y 5 el más alto.

A continuación se va a medir la distorsión de 3 imágenes con distintas resoluciones y diferentes cantidades de información introducida:

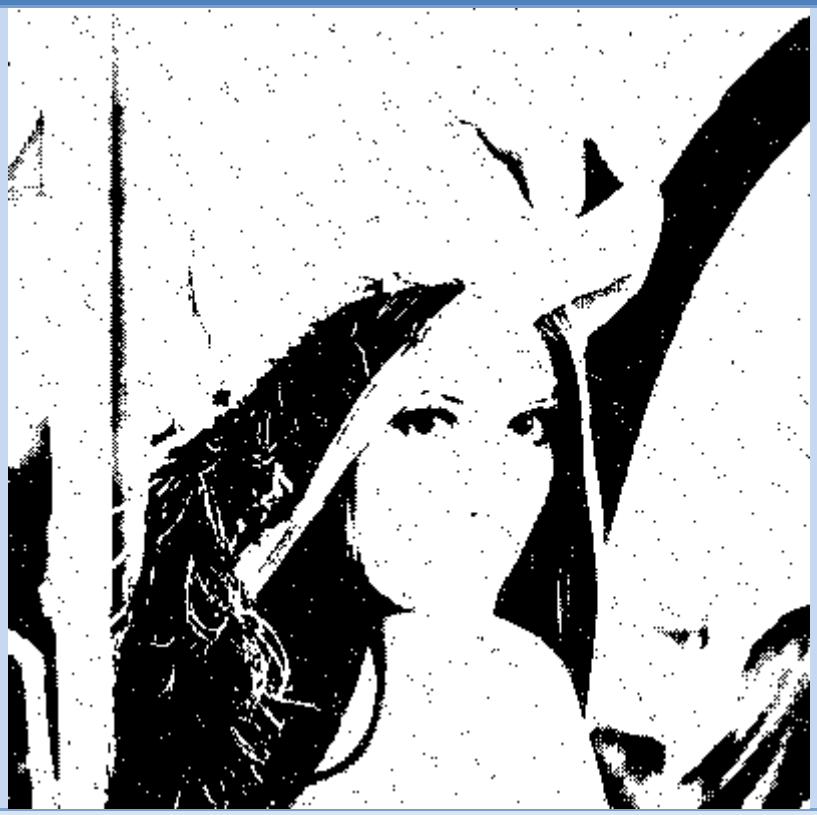
		
Pimientos	**Lena**	**Mandril**
0.0108 bpp	**0.1008 bpp**	**0.4902 bpp**
PSNR 32.044 dB	**PSNR 30.678 dB**	**PSNR 30.34 dB**

Tabla 121: Imágenes esteganografiadas obtenidos tras una ocultación sin compresión.

En la tabla anterior se observa que la imagen que refleja un índice de distorsión mayor es la tercera, la del mandril, en la que se han ocultado 1000 caracteres de información. La calidad subjetiva de esa imagen y la de Lena, en la que se han ocultado 500 caracteres, son consideradas según la tabla 120 como regulares, sin embargo visualmente no se aprecia una repercusión visual importante. La primera imagen, la de los pimientos tiene un coeficiente de PSNR mayor, lo que indica que tiene menos distorsiones internamente. Esto se debe a que sólo se introdujeron 100 caracteres en su interior. Su calidad subjetiva es considerada como buena.

A continuación se examinará una de las 3 imágenes anteriores para analizar sus patrones esteganográficos con mayor profundidad. Se usará la imagen de Lena para este análisis. En primer lugar se desglosarán los planos de bits que componen la figura esteganografiada. Como la información oculta en este modo se almacena en los 3 bits menos significativos de los píxeles se analizarán únicamente estos.

Plano de bit 3



Se aprecia una gran cantidad de ruido en la imagen en forma de píxeles blancos y negros. La repercusión visual en la imagen esteganografiada no era apreciable, sin embargo en este corte del tercer plano de bit se hace muy evidente la existencia del mensaje oculto.

Plano de bit 2

Al igual que en la imagen anterior es muy notorio el ruido existente en el segundo plano de bit.

Plano de bit 1

Este plano es el que presenta mayor aleatoriedad de todos ya que es el nivel más interno de la imagen. Sin embargo se sigue apreciando puntualmente la presencia de píxeles aislados cuyo color es distinto del de los adyacentes.

Tabla 122: 3 últimos planos de bits de la imagen Lena.jpg.

Se aprecia nítidamente cómo en los planos de bit 3 y 2 (el primero y segundo de la tabla 122) el ruido correspondiente al mensaje oculto se encuentra en los mismos píxeles. Esto es debido a que la inserción de los 3 bits de redundancia se realiza en el mismo byte consecutivamente. De hecho también se aprecia en el plano de los bits menos significativos la misma coincidencia, sin embargo este plano también contiene la información no redundante insertada en el LSB de otros bytes distintos.

Para finalizar se ha empleado la herramienta *Stego-Master* para analizar algunos patrones estadísticos de la imagen esteganografiada de Lena. Los datos obtenidos son empleados por muchos programas para determinar si una imagen porta información oculta o no:

	Imagen de Lena original	Imagen de Lena con datos
Transiciones de bit (%)	29.76%	16.06%
Bits a 1 en el LSB (%)	50.06%	51.05%
Varianza en el LSB	119999	119892

Tabla 123: Medidas esteganográficas de Lena.

Se observa que disminuye significativamente el porcentaje de transiciones de bit y la medida de la varianza, por otro lado aumenta en un 1% la cantidad de bits con valor 1 en el bit menos significativo de los bytes de la imagen. Todo ello representa una disminución de la aleatoriedad en la imagen esteganografiada, de modo que las herramientas que miden este patrón podrían detectar nuestra imagen portadora como tal.

ROBUSTIDAD

Generalmente la imagen esteganografiada no sufre ningún tipo de modificación, como es la compresión, en un canal de transmisión ideal, de modo que es posible siempre recuperar el mensaje oculto. Sin embargo, cuando dicho canal no es perfecto se puede introducir ruido en el medio portador destruyendo el mensaje oculto. La medida que evalúa la recuperación de errores en el mensaje oculto se llama **BER (Bit Error Rate)**.

El BER es la medida más comúnmente empleada para determinar la cantidad de errores durante la transmisión que se pueden producir, y se define como la posibilidad de que un bit se reciba erróneamente. Se obtiene del cociente de dividir los bits erróneamente transmitidos entre el total de bits enviados.

A continuación se va a medir la cantidad de errores que se producen durante el envío por email y MMS de las imágenes esteganografiadas de la tabla 124:

Imagen enviada	Email	MMS
Pimientos	0 errores	0 errores
Lena	0 errores	0 errores
Mandril	0 errores	0 errores

Tabla 124: Errores en la recepción de imágenes sin compresión.

Se obtiene el resultado esperado en todos los casos ya que el envío por estos medios no supone ninguna modificación del medio. La única compresión que sufren es antes del envío cuando se convierten en formato PNG (sin pérdidas), no se introduce ninguna interferencia en el medio. El valor del BER en todos los casos es 0. Sin embargo no se descarta que puntualmente se produzca algún error de transmisión, en cuyo caso podrán ser corregidos mediante los códigos de corrección de errores del que dispone nuestro sistema.

Posteriormente veremos que tras la transmisión de las imágenes por medios que imponen compresión como WhatsApp, Line o Spotbros sí se producen errores que será necesario corregir para la correcta recuperación del mensaje completo.

CONCLUSIONES

El método esteganográfico de ocultación sin compresión ha funcionado en todas las pruebas realizadas, no sólo en las descritas en la presente memoria sino enviando y recibiendo imágenes esteganografiadas de otros compañeros con la aplicación instalada.

Las 3 características principales de un sistema esteganográfico guardan un gran equilibrio para este método, ya que:

- La capacidad del mensaje es suficiente para el uso que se le va a dar, sin embargo queda abierta la posibilidad de aumentar el número de caracteres permitidos.
- La detección de un mensaje oculto se produce cuando se emplean técnicas de estegoanálisis avanzado en las que se toman medidas como las reflejadas en la tabla 123, y cuando se analizan los planos de bits que componen la imagen detenidamente.
- La robustez empelada permite recuperar bits mal transmitidos del mensaje.

7.1.2 OCULTACIÓN CON COMPRESIÓN DE IMAGEN

Se evaluarán las propiedades esteganográficas descritas en la sección anterior para el subsistema de ocultación que requiere compresión, que nos permite enviar imágenes con mensajes ocultos a través de WhatsApp, Line y Spotbros.

CAPACIDAD

Al igual que en el subsistema anterior, sólo se pueden ocultar 1000 caracteres en una imagen. Se hallará el número de bits por píxel (bpp) para medir la capacidad de este método. Se utilizará una imagen de 800x600 píxeles como referencia para obtener el valor bbp en distintos casos. En primer lugar vamos a desglosar el número de bits que se ocultan en este modo.

El algoritmo de ocultación con compresión utiliza:

- **44 Bits de longitud**.

  Como entrada se tienen 32 bits de longitud, que son convertidos en primer lugar en códigos de redundancia. Por cada byte ocultado en este modo se genera un código *Hamming* de redundancia de 11 bits. Por tanto estamos hablando de $4 \times 11 = 44$ bits que representan la longitud. Teniendo en cuenta que cada uno de esos bits se esconde en un píxel completo, se modifican un total de 44 píxeles, de lo que se obtiene una cantidad de $44 \times 24 = 1056$ bits totales modificados en la imagen. El 24 se obtiene del número de bits que conforman un píxel (8 bits por cada una de las componentes RGB).

- **11 Bits para ocultar cada carácter cifrado**.

  Al igual que ocurre con la longitud, es necesario convertir los 8 bits que representan cada carácter en 11 de redundancia. Se obtiene que el mensaje ocupa un total de $L \times 11 \times 24$ bits en la imagen, donde L es la longitud del mensaje, 11 la cantidad de bits que ocupa la redundancia del carácter, y 24 el número de bits que componen un píxel.

  Se tomarán 5 valores de ejemplo como longitud del mensaje, de manera que podamos obtener la cantidad de bits por píxel en cada caso, todo ello sobre una imagen de 800x600 píxeles:

Longitud del mensaje	Bits del mensaje	Bits totales con longitud	Bits por píxel (bpp)
20 caracteres	5280	6336	**0.0132**
50 caracteres	13200	14256	**0.0297**
100 caracteres	26400	27456	**0.0572**
500 caracteres	132000	133056	**0.2772**
1000 caracteres	264000	265056	**0.5522**

Tabla 125: Bits por píxel para el algoritmo de ocultación con compresión.

Se observa que las medidas para el bpp obtenidas en cada caso son 8.25 veces superiores a las calculadas en la sección anterior para el algoritmo sin compresión. Esto sucede porque la cantidad de redundancia que se introduce en este modo es mucho más elevada.

Es necesario mencionar que toda inserción de información requiere un número mínimo de píxeles disponibles en la imagen, de manera que si no se satisface esa cantidad no podrá ocultarse el mensaje. En la siguiente tabla se muestra la cantidad de píxeles requeridos para insertar información:

Longitud del mensaje	Píxeles necesarios	Tamaño de imagen
20 caracteres	264	17x17 píxeles
50 caracteres	594	25x25 píxeles
100 caracteres	1144	34x34 píxeles
500 caracteres	5544	75x75 píxeles
1000 caracteres	11044	106x106 píxeles

Tabla 126: Píxeles requeridos para la inserción de distintas longitudes con compresión.

En la tabla anterior se puede comprobar que se requieren menos píxeles y, por tanto, imágenes más pequeñas para ocultar el mismo número de caracteres que el método sin compresión. Esto sucede porque mientras que el algoritmo anterior requiere 16 píxeles para ocultar cada carácter en los LSB, éste necesita 11 píxeles por carácter para hacerlo. La dispersión es más baja en este caso (los bits modificados se almacenan en píxeles completos y no en los últimos bits de los bytes de la imagen como ocurría con el algoritmo sin compresión).

Se considera que la capacidad máxima del mensaje limitada en 1000 caracteres es sobradamente suficiente para introducir un mensaje secreto. A diferencia del método anterior en el que abríamos la posibilidad de aumentar la capacidad del sistema, en este caso no se hará porque la repercusión visual sería inmediatamente perceptible.

INDETECTABILIDAD

El técnica esteganográfica empleada para este algoritmo, como se comentó anteriormente, introduce el mensaje modificando el valor completo de los píxeles según proceda. En muchos casos el bit a introducir puede ser recuperado sin necesidad de variar el valor del píxel, en otros es necesario modificarlo completamente, produciendo las repercusiones visuales que se mostrarán en los siguientes casos.

A continuación se va a medir la distorsión en 3 imágenes de distintas resoluciones. Se han escogido 3 imágenes que tienen una resolución grande (800x600 píxeles) para que la repercusión visual no sea muy elevada. La tarea de elección de la imagen que va a ocultar el mensaje en este modo es muy importante, las imágenes escogidas deberían:

- Tener una resolución grande, se recomienda el tamaño 800x600.
- Tener una variabilidad de colores en su interior grande, de modo que el ruido introducido sea lo más imperceptible posible.
- Portar la menor información posible, introducir mensajes de 100 caracteres como máximo en las imágenes es una práctica muy recomendada en este modo, pues cuantos menos caracteres introduzcamos menor será la repercusión visual final.

Las imágenes están obtenidas tras ser enviadas por WhatsApp, son las siguientes:

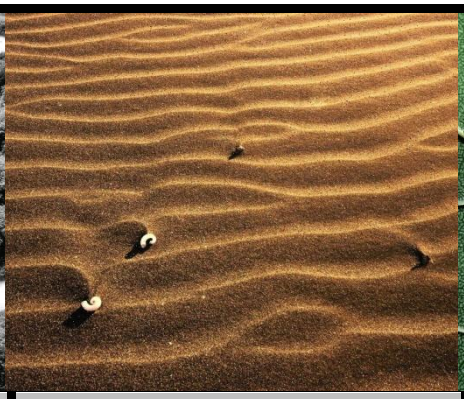
		
Rocas	**Arena**	**Hoja**
0.0572 bpp	**0.2772 bpp**	**0.5522 bpp**
PSNR 25.06 dB	**PSNR 23.57 dB**	**PSNR 20.95 dB**

Tabla 127: Imágenes esteganografiadas obtenidos tras una ocultación con compresión.

Cuantos más caracteres se oculten en este modo peor va a ser la calidad resultante de la imagen final. Hay que tener presente que estas imágenes han sufrido una compresión JPEG muy agresiva, que representa el mayor factor de distorsión de la imagen. Las imágenes anteriores se caracterizan por la variabilidad de texturas y formas, lo cual es beneficioso para no ser muy perceptible. En la primera imagen, la de las rocas, se han ocultado 100 caracteres, su calidad resultante es calificada por la tabla 120 como regular. La segunda y tercera imágenes ocultan 500 y 1000 caracteres respectivamente, y ambas son calificadas de mala calidad. A pesar ello se han examinado en profundidad dichas imágenes y no se consideran inaceptables para haber sido transmitidas a través de mensajería instantánea.

A continuación se examinará la imagen “Hoja.jpg” que ha sido mostrada en la tabla 127. En primer lugar se desglosarán los planos de bits que componen la figura. La información ocultada en este modo se puede insertar en todos los bits necesarios de un píxel, esto quiere decir que las modificaciones están repartidas heterogéneamente por la imagen. Visualmente sólo será perceptible la información oculta en los planos de bits más significativos, ya que son estos los que contribuyen en mayor medida a la tonalidad de la imagen. El resto de planos presentará un patrón bastante aleatorio y será difícil apreciar detalles significativos.

En primer lugar se mostrará la imagen esteganografiada y bajo ella el plano número 8, que representa los bits más significativos de la imagen (MSB). En último lugar se mostrará en menor tamaño la descomposición de los planos 8 al 1.

Imagen esteganografiada

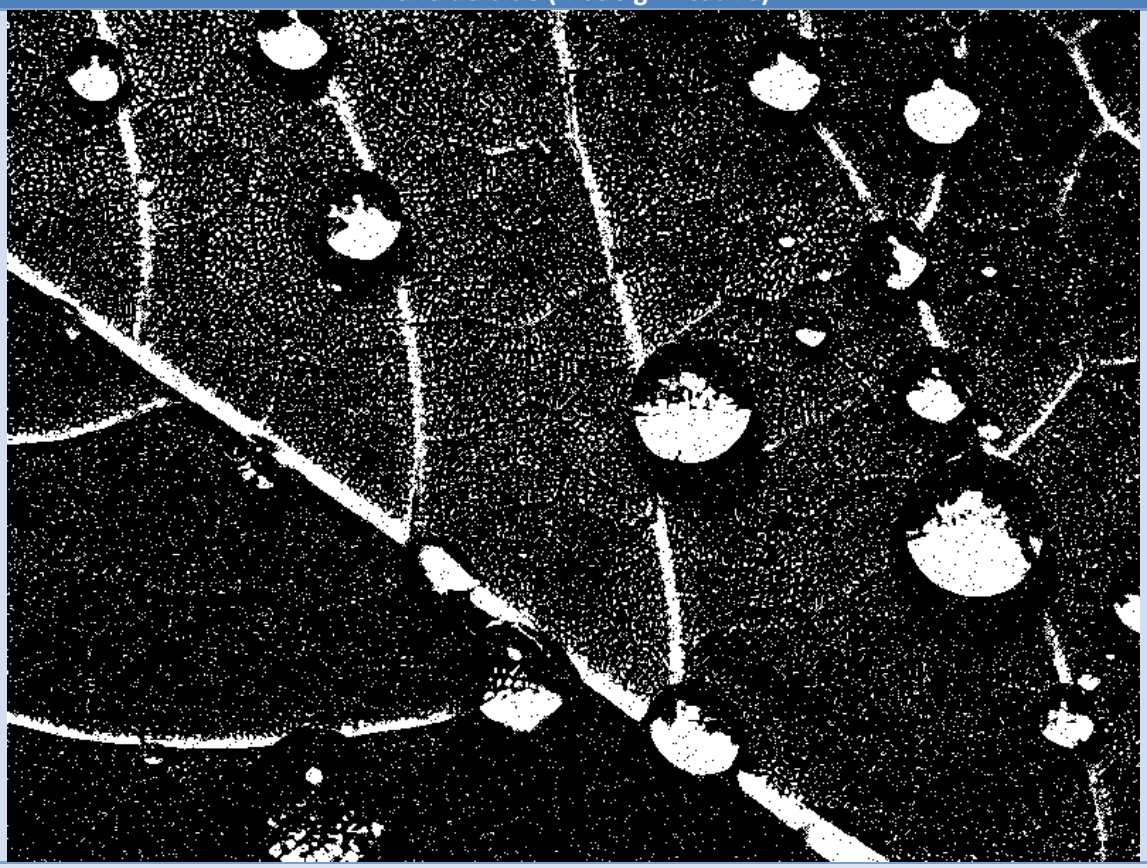
Plano de bit 8 (Más significativo)

Se observan muy fácilmente los píxeles modificados en la parte inferior de la imagen y en las gotas.

Tabla 128: Comparativa de la imagen esteganografiada y del plano de bits más significativo.

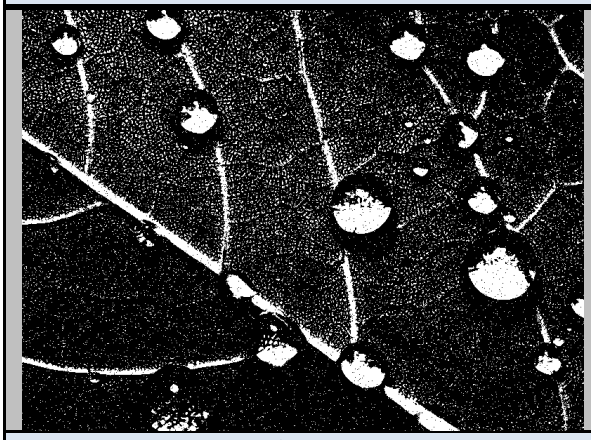
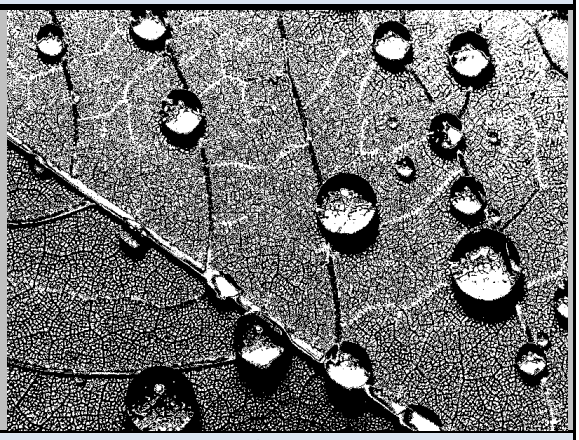
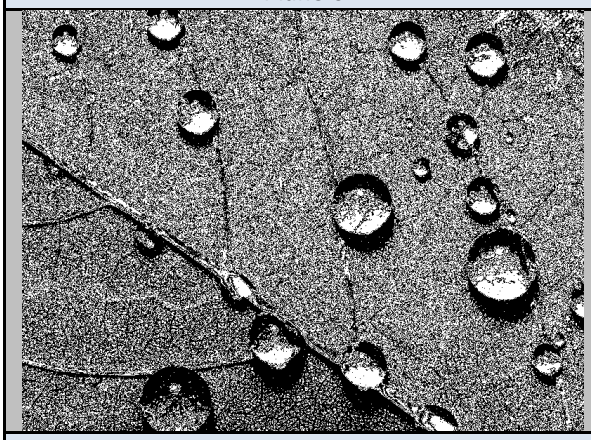
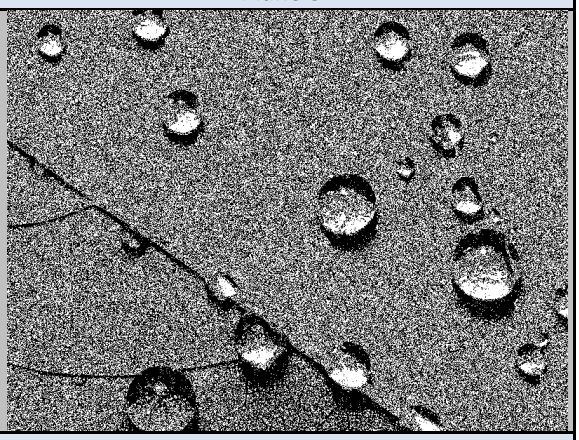
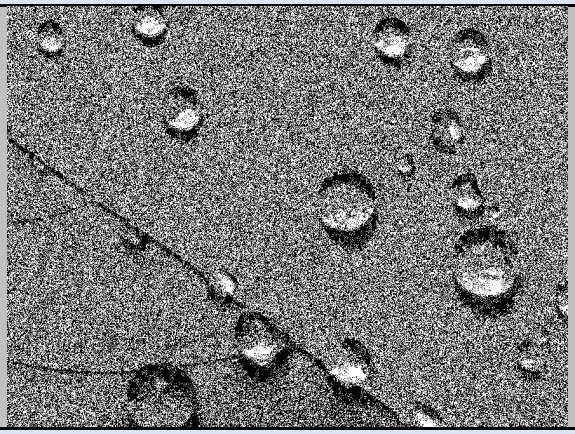
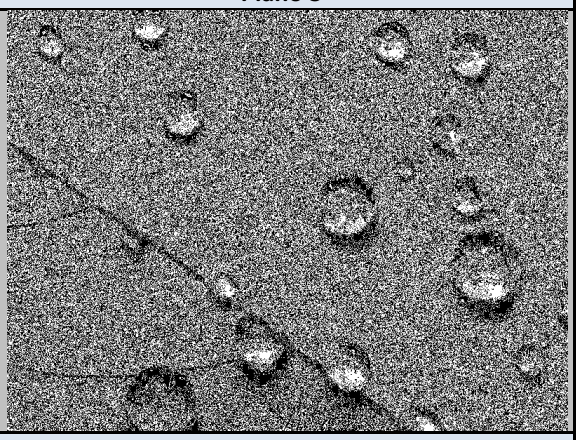
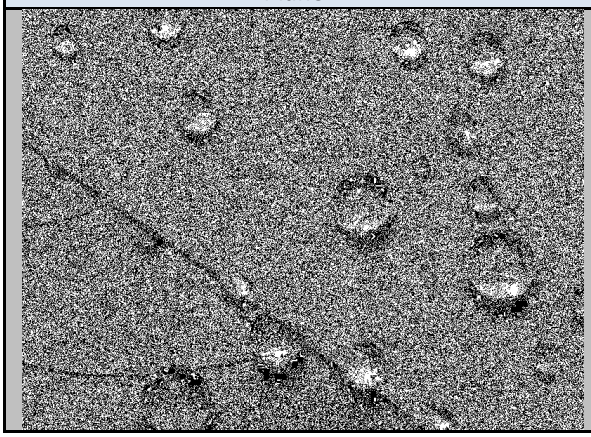
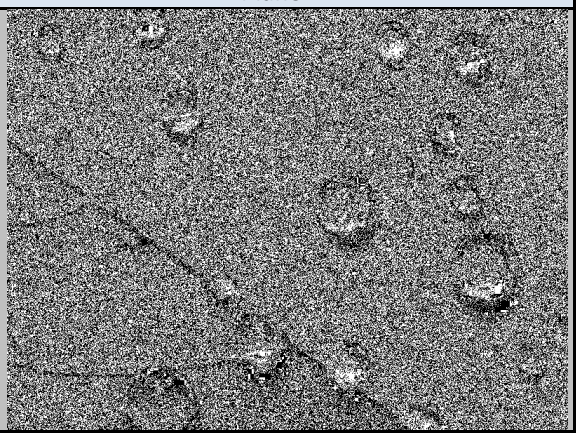
Plano 8	Plano 7
	
Plano 6	**Plano 5**
	
Plano 4	**Plano 3**
	
Plano 2	**Plano 1**
	

Tabla 129: Desglose de planos de bits para imagen Hoja.

Como se aprecia en la tabla 128, el plano de bits más significativo contiene la mayoría de ruido y repercusión visual de la imagen original, ya que en él es donde produce los cambios más significativos el algoritmo de ocultación. En la tabla 129 se desglosan el resto de planos, se pueden apreciar los artefactos visuales hasta el plano número 5, a partir de éste hasta el plano menos significativo el ruido natural de la imagen no dejan entrever ningún patrón reseñable.

Para finalizar este punto se ha comprobado con varios programas de estegoanálisis las imágenes de la tabla 127. A continuación se muestran los resultados obtenidos:

Programa	Rocas.jpg	Arena.jpg	Hoja.jpg
Stegdetect	JPHIDE(*)	No detectado	No detectado
StegSecret	No detectado	No detectado	No detectado
StegSpy	HIDERMAN	No detectado	No detectado

Tabla 130: Tabla de detección con herramientas de estegoanálisis.

Las 3 herramientas de estegoanálisis que se han usado para el análisis de la tabla 130 son comúnmente utilizadas en el mundo de la esteganografía. La imagen Rocas.jpg, que contenía 100 caracteres ocultos, es detectada por los programas *Stegdetect* y *StegSpy*, sin embargo cada uno de ellos apunta a un programa de ocultación distinto (*JPHIDE* y *Hiderman*). La detección se realiza mediante búsqueda de firmas, heurísticas y patrones estadísticos, se cree que ha sido detectado por el segundo procedimiento. Las heurísticas buscan comportamientos sospechosos en las imágenes que puedan indicar que se oculta un mensaje en ellos. De hecho, en muchas ocasiones se producen falsos positivos con imágenes que no tienen información oculta. Curiosamente las imágenes que contenían más información oculta (Arena.jpg y Hoja.jpg) no han sido detectadas como portadoras por los programas anteriores.

Para hacernos una idea del porcentaje de imágenes generadas que son detectadas, se han analizado 100 imágenes distintas esteganografiadas con nuestra aplicación con contenidos de diferentes tamaños. El resultado obtenido indica que el 15% de las fotos son detectadas como imágenes generadas por software esteganográfico. Esto quiere decir que el 85% restante goza de total indetectabilidad ante los programas empleados.

ROBUSTIDAD

Las imágenes que se envían a través de WhatsApp, Line o Spotbros están expuestas a modificaciones, escalados y compresiones durante su transmisión a través de los protocolos de dichas aplicaciones. Por ello se decidió introducir un código de 11 bits de corrección de errores por cada byte introducido usando *Hamming*.

Ahora se va a medir la cantidad de errores que se corrigen tras su recepción para extraer el mensaje oculto. En la siguiente tabla se muestra el número de errores corregidos por la aplicación para las imágenes de la tabla 127 usando los 3 modos de envío descritos:

Imagen enviada	WhatsApp	Line	Spotbros
Rocas.jpg	0 errores	3 errores	0 errores
Arena.jpg	0 errores	3 errores	37 errores
Hoja.jpg	2 errores	7 errores	No se puede

Tabla 131: Errores en la recepción de imágenes con compresión.

En la tabla anterior se obtienen los resultados esperados para los envíos a través de WhatsApp y Line, en los que la recuperación de errores funciona perfectamente y es posible mandar mensajes de hasta 1000 caracteres sin problemas, aunque es desaconsejable.

En el caso de Spotbros los resultados no son completamente satisfactorios por su irregularidad. Se han hecho muchas pruebas adicionales con esta aplicación de mensajería instantánea y en algunas ocasiones somos capaces de enviar y recibir mensajes de 1000 caracteres y otras veces no. La razón de obtener resultados tan dispares se debe a que el parámetro de compresión en Spotbros es dinámico, como se vio en la sección de ingeniería inversa, y por tanto unas imágenes se comprimen con un ratio de compresión y otras con otro parámetro distinto. En todo caso, siempre que se oculten mensajes cortos debería funcionar sin problemas.

A pesar de que se hayan producido errores durante la transmisión de la información la aplicación es capaz de recuperar el mensaje ocultado en la gran mayoría de los casos, por tanto se afirma que la robustez del algoritmo esteganográfico es alta.

CONCLUSIONES

El método esteganográfico de ocultación con compresión ha funcionado en la gran mayoría de pruebas realizadas, tan solo se han obtenido fallos puntuales al intentar enviar mensajes de gran tamaño (500 caracteres o más) a través de Spotbros.

Respecto a las 3 propiedades fundamentales de los sistemas esteganográficos que posee el sistema de ocultación con compresión, se puede resumir que:

- La capacidad del mensaje es la misma que en el sistema sin compresión, 1000 caracteres como límite, pero se aconseja encarecidamente que se empleen siempre que sea posible menos de 100, pues con más podrían producirse errores (como ocurre con Spotbros puntualmente).
- La detección de un mensaje oculto se produce en un 15% de los casos. Es fundamental escoger una imagen portadora adecuada para evitar este tipo de detecciones.
- La robustez juega un papel crucial en este sistema porque deben corregirse los errores que se produzcan durante la transmisión.

El atributo más sacrificado ha sido la perceptibilidad de los píxeles ocultados, pues si no fueran perceptibles el sistema no sería capaz de sobreponerse a una compresión como la que tiene lugar con las aplicaciones de mensajería analizadas.

7.2 RENDIMIENTO DE LA APLICACIÓN

En esta sección se va a analizar el rendimiento global de la aplicación. Este rendimiento se traduce en la velocidad con la que es capaz de ocultar y extraer los mensajes ocultos.

Las pruebas siguientes se van a realizar en un dispositivo móvil de 1GHz de procesador, cuyo rendimiento es aceptable dentro del contexto tecnológico actual. En primer lugar se medirá la velocidad de ejecución del modo de ocultación y posteriormente la de extracción.

7.2.1 RENDIMIENTO MODO DE OCULTACIÓN

Para medir los tiempos se han realizado 20 ejecuciones con distintos ficheros de imagen y mensajes ocultos, después se ha realizado la media aritmética de los valores obtenidos.

El modo de ocultación para WhatsApp, Line y Spotbros es bastante más lento que el que no requiere compresión y es usado con email y MMS. Los resultados obtenidos son:

- El modo de ocultación con compresión requiere aproximadamente entre 11 y 15 segundos para su correcta ejecución, lo que devuelve una media de 13 segundos en el caso de que no haya que reintentar la ocultación. En la siguiente tabla se ha medido el tiempo que conlleva la ejecución para cada reintento con una imagen de medianas dimensiones (600x400):

Reintentos	Tiempo total
0	8s
1	15s
2	23s
3	30s
4	37s
5	45s

Tabla 132: Tiempo de ejecución para imagen mediana.

En el caso de que la imagen tenga dimensiones grandes (800x600) se requeriría el siguiente tiempo de media:

Reintentos	Tiempo total
0	18s
1	35s
2	50s
3	68s
4	75s
5	90s

Tabla 133: Tiempo de ejecución para imagen grande.

Se observa que los tiempos aumentan drásticamente llegando a requerir en el peor de los casos de 90 segundos para ocultar el mensaje. En muy pocos casos se debería llegar a ese extremo siendo lo más habitual que la ocultación se produjera sin reintentos en una media de 13 segundos.

No obstante ese tiempo máximo de 90 segundos se reducía a 30 si el dispositivo móvil en que se ejecutase tuviera 1.2GHz, es decir, sólo 200MHz más que el actual.

- El modo de ocultación sin compresión consume de media 4.5 segundos para cualquier tamaño de imagen. La razón por la que los tiempos para un modo y otro son tan dispares es porque el primero tiene que realizar varias compresiones y descompresiones durante su ejecución, y el segundo simplemente oculta el mensaje y guarda la imagen.

Se han obtenido las medidas del tiempo requerido por los principales métodos del sistema, a continuación se presentan aquellos que se han considerado más significativos:

Método	Tiempo en ms
Obtener mapa de dispersión	10ms
Escalado de imagen	70ms
Convertir imagen a bytes	150ms
Descifrado del mensaje	320ms
Cifrado del mensaje	350ms
Convertir bytes a imagen	760ms
Compresión de imagen	1600ms

Tabla 134: Tiempo de ejecución de los métodos principales.

7.2.2 RENDIMIENTO MODO EXTRACCIÓN

Las pruebas realizadas para el módulo de extracción han revelado que el tiempo medio que se tarda en extraer un mensaje de una imagen es de 1.5 segundos. Este valor se obtiene independientemente de que la imagen sea grande o pequeña. Es bastante rápido comparado con el tiempo que se consume en la ocultación del mensaje.

Obviamente todas las mediciones de tiempo realizadas se agilizarían con un dispositivo móvil con mayores prestaciones, asimismo se ralentizaría en uno que tuviera menos. En todo caso el requisito de que el sistema tenga un tiempo de ejecución menor a 1 minuto se cumple en la gran mayoría de las ocasiones.

7.3 SEGURIDAD DE LA APLICACIÓN

El proceso de seguridad está presente en cada una de las fases y módulos que componen la aplicación. Para evaluar las medidas de seguridad adoptadas se proponen las siguientes pruebas sobre el sistema:

- Comprobar la fortaleza de los algoritmos de cifrado empleados.
- Realizar un escaneo de la memoria del dispositivo móvil para obtener las contraseñas y mensajes escritos en la aplicación.
- Realizar ingeniería inversa sobre el programa para obtener detalles internos del mismo.

7.3.1 FORTALEZA DEL CIFRADO EMPLEADO

Como se describió en el capítulo de implementación, el cifrado que emplea la aplicación es AES con una clave de 256 bits. Este algoritmo no está considerado roto, el único ataque posible es la búsqueda exhaustiva de claves o fuerza bruta, se trata de probar todas las combinaciones posibles de la clave hasta dar con la correcta que descifre el mensaje. Esto no es tarea fácil para un atacante pues los recursos computacionales actuales no son suficientes como para hacer viable el ataque. A continuación se muestra en la siguiente tabla el número de posibles combinaciones para cada longitud de clave:

Tamaño de clave	Posibles combinaciones
1 bit	2
2 bits	4
4 bits	16
8 bit	256
16 bits	65536
32 bits	4.2×10^{9}
56 bits (DES)	7.2×10^{16}
64 bit	1.8×10^{19}
128 bits (AES)	3.4×10^{38}
192 bits (AES)	6.2×10^{57}
256 bits (AES)	$\mathbf{1.1 \times 10^{77}}$

Tabla 135: Relación entre tamaño de clave y las combinaciones que supone [27]

Encontrar una clave de 256 bits por fuerza bruta requeriría 2^{128} veces más capacidad computacional que para una clave de 128 bits. Esto quiere decir que si, por ejemplo, se tuvieran 50 superordenadores capaces de comprobar 10^{18} claves AES por segundo contra nuestro sistema, se tardaría teóricamente para recorrer todas las claves posibles 3×10^{51} años, lo cual es obviamente inviable [28].

La principal debilidad del sistema criptográfico y esteganográfico es el usuario, el eslabón más débil de toda cadena de seguridad, ya que de él depende la elección de la contraseña que se encargará del cifrado y la ocultación del mensaje.

7.3.2 ANÁLISIS DE MEMORIA

La aplicación desarrollada no almacena ningún tipo de información sobre el usuario en el móvil, el único contenido que genera son las imágenes esteganografiadas. No se crean bases de datos ni estructuras permanentes que puedan comprometer las credenciales del usuario.

También se ha tenido en cuenta la posibilidad de que un atacante tuviera acceso al móvil de la víctima y pudiera realizar un análisis de la memoria volátil del dispositivo. En muchos casos esta técnica de adquisición de información tiene muy buenos resultados pues se pueden ver las variables que han sido usadas por algunas aplicaciones, que en algunos casos corresponden con información sensible del usuario como son sus contraseñas.

La técnica anterior no obtendría fruto en el caso de nuestro sistema ya que una vez se ha empleado la contraseña para cifrar el mensaje y obtener el mapa de dispersión, se sobrescribe su valor con información aleatoria. Lo mismo ocurre con el mensaje ocultado que, tras haberse cifrado, se sustituye su contenido por otro valor escogido al azar. De este modo nos aseguramos de que un escaneo de memoria no sea capaz de recuperar los datos mencionados.

Se han realizado pruebas en tiempo de ejecución con el "analizador de memoria" que tiene Eclipse para Android, y la única forma que tendría un atacante de obtener la contraseña sería analizando la memoria del dispositivo en el instante justo anterior a que sea sobrescrita dicha variable. La única información útil que se puede obtener de la memoria es el vector de inicialización original para generar el *salt* durante el proceso de cifrado.

7.3.3 ATAQUE DE INGENIERÍA INVERSA

Durante el desarrollo del actual proyecto se ha tenido que recurrir a herramientas que son capaces de convertir una aplicación ejecutable en código fuente. Sin embargo, como hemos visto en los casos de WhatsApp, Line y Spotbros el código fuente extraído no era completamente inteligible, estaba ofuscado, lo que complicó nuestra tarea para obtener los parámetros de compresión y otros valores como el escalado de las imágenes.

La ofuscación de código realiza las siguientes tareas:

- Elimina el código no usado, esto incluye comentarios y anotaciones.
- Renombra las clases, métodos, variables y parámetros con nombres ofuscados.

De manera que el objetivo de ofuscar el código fuente de un programa es dificultar la tarea de las herramientas de ingeniería inversa.

La aplicación desarrollada ha empleado técnicas de ofuscación de código para evitar que los atacantes tengan acceso directo a los algoritmos de cifrado y de ocultación. Se trata de llegar a la **seguridad por la oscuridad** en este caso. Es cierto que los algoritmos son lo suficientemente seguros para no temer por su conocimiento, pero siempre viene bien introducir una capa más de seguridad para complicar el trabajo de los atacantes. Se ha empleado la utilidad *ProGuard* para ofuscar el código del programa tras su compilación.

7.4 PUBLICACIÓN DE LA APLICACIÓN

Tras meses de trabajo en la aplicación, en los que se implementaron todas las funcionalidades que han sido descritas en la presente memoria, se tomó la decisión con apoyo del tutor de subir el programa a ***Google Play*.**

Google Play es la tienda de software en línea principal para los dispositivos Android, actualmente cuenta con más de 400.000 aplicaciones. Las motivaciones para subir la aplicación fueron:

- Comprobar si el software tenía una utilidad real para los usuarios.
- Medir el grado de satisfacción por parte de los usuarios.
- Comprobar la compatibilidad de la aplicación en todo tipo de dispositivos Android.
- Corregir errores tras el reporte de incidencias.

7.4.1 PROGRESIÓN Y DESCARGAS

La aplicación fue subida a *Google Play* el día 16 de Diciembre de 2012 bajo el nombre original de "Mensajes secretos para WhatsApp". Inicialmente la aplicación tenía la funcionalidad de ocultar mensajes secretos en imágenes para enviarlos por WhatsApp y email.

Después, tras comprobar la gran cantidad de descargas que obtuvo en la primera semana (más de 3000), se decidió hacer la aplicación compatible con Line, un sistema de mensajería instantánea que tiene cada día más usuarios.

Tras esta actualización las descargas se fueron incrementando progresivamente. Hasta alcanzar a día 31 de Diciembre un número de descargas superior a 9000. En Enero se decidió ampliar el sistema incorporando la posibilidad de enviar imágenes esteganografiadas a Spotbros y a través de MMS. El nombre de la aplicación fue cambiado a "**Mensajes secretos para Android**" ya que la funcionalidad no se limitaba sólo a WhatsApp.

En el siguiente gráfico se contempla la evolución del número de descargas totales de la aplicación:

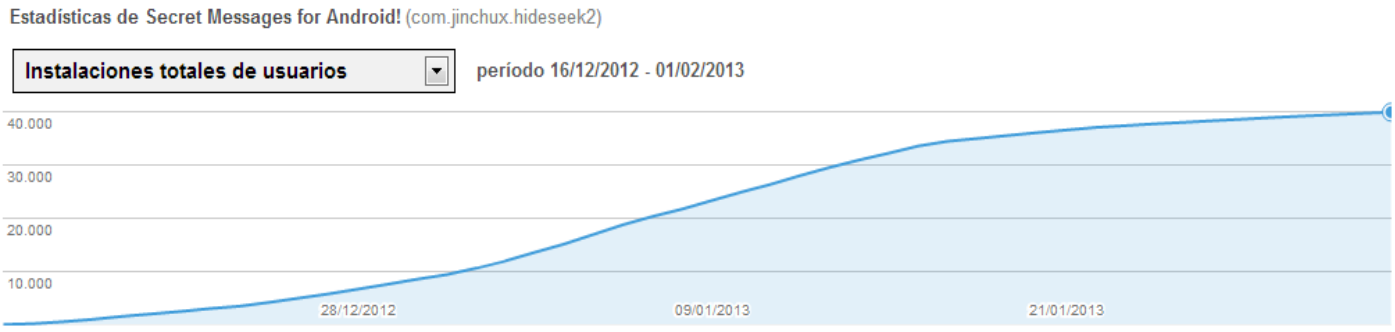


Ilustración 53: Descargas totales de la aplicación.

Hasta el día 1 de Febrero se han realizado más de **40000 descargas de la aplicación**. El incremento de descargas fue constante hasta el día 16 de Enero, cuando cumplía un mes, a partir de entonces el ritmo se estabilizó y bajo considerablemente hasta el día de hoy.

En el siguiente gráfico se muestra la evolución de las descargas diarias a lo largo del periodo:

Ilustración 54: Descargas diarias de la aplicación.

Se puede observar que el día 5 de Enero se produjo el pico más alto de descargas diarias de la aplicación, se produjeron en solo un día 1798 descargas de distintos usuarios. A día de hoy tan solo se producen unas 400 nuevas descargas diarias.

Los resultados obtenidos a partir del número de descargas de los usuarios demuestra la utilidad real que ha tenido la aplicación.

7.4.2 OPINIONES DE LOS USUARIOS

Hasta día de hoy se han realizado 55 opiniones de distintos usuarios sobre la aplicación. La puntuación media de la aplicación es de 3.7 puntos sobre 5, un valor bastante aceptable considerando el número de descargas obtenidas. En la siguiente figura se muestran las opiniones de los usuarios:

Ilustración 55: Opiniones de los usuarios.

A través de las opiniones negativas y correos electrónicos recibidos de los usuarios se ha llegado a las siguientes conclusiones:

- Algunos usuarios muestran síntomas de no saber utilizar la aplicación.
- Otros alegan que la aplicación no funciona bien.

Para solucionar el primer problema se decidió hacer un video-tutorial explicativo de cómo se usa el programa y adjuntarlo a la descripción principal. Para el segundo problema se analizaron los dispositivos en los que se había tenido problemas de ejecución y se verificó si efectivamente eran o no compatibles. Una vez se conocían los terminales en los que no funcionaba la aplicación se restringió la descarga en los mismos usando el sistema de filtrado que ofrece *Google Play*.

7.4.3 INCIDENCIAS

Cuando la aplicación tiene un problema durante la ejecución en un dispositivo se produce un fallo y se cierra el programa. Los errores que se producen son reportados automáticamente por Google, de manera que desde el panel de administración de la aplicación puedo ver qué problemas de funcionamiento ha tenido la aplicación y en qué modelos de dispositivos y versiones de Android se han producido.

Tras el análisis de más de 168 informes de error se han podido arreglar muchos de los problemas reportados. El error más repetido es el relacionado con la **insuficiencia de memoria**, que se ha reportado hasta en 126 ocasiones distintas. Este problema puede deberse a la limitada capacidad de memoria que tienen algunos dispositivos Android. No es un problema de solución sencilla, a pesar de ello se ha depurado minuciosamente el código para intentar realizar el menor uso de memoria posible liberando estructuras temporales y gestionando los recursos de la manera más óptima.

Han sido de gran ayuda todas estas notificaciones de problemas, ya que han hecho que se puedan localizar los problemas fácilmente para corregirlos.

7.4.4 RESULTADOS OBTENIDOS

Los buenos resultados obtenidos tras la publicación de la aplicación indican el éxito del proyecto. La cantidad de descargas producidas se debe en parte a la originalidad del software, ya que no existe actualmente ninguna otra aplicación que permita el envío de imágenes esteganografiadas a través de sistemas de mensajería instantánea que exigen compresión, como son WhatsApp, Line y Spotbros.

Además, de todas las aplicaciones de esteganografía que hay disponibles para Android, la que se ha desarrollado es la que ha tenido más éxito a pesar de no llevar ni dos meses publicada. Basta con realizar una búsqueda en *Google Play* en la que se usen las palabras “mensajes secretos” o “esteganografía” entre otras para que aparezca la aplicación entre las primeras.

De hecho el día 16 de Enero la aplicación llegó a situarse en el puesto 40 de la tabla de aplicaciones líderes de comunicación en España. Desde otras páginas como *www.androidexcellence.com* se han realizado análisis de la aplicación, donde se califica como “*una aplicación muy útil que proporciona una seguridad muy buena*” [29].

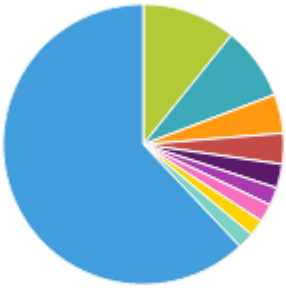
El país desde el que se han producido más descargas ha sido Arabia Saudí, seguido muy de cerca de España y México. Una de las razones principales que han facilitado la distribución geográfica de la aplicación ha sido la traducción al **inglés** que se realizó. Entre los países que se han registrado como más activos descargando la aplicación se destacan los mostrados en la figura 56.

Instalaciones totales de usuarios el 02/02/2013

Arabia Saudí	7.931	19,74 %
España	7.645	19,03 %
México	5.557	13,83 %
Omán	2.165	5,39 %
India	1.644	4,09 %
Líbano	1.303	3,24 %
Argentina	1.042	2,59 %
Estados Unidos	987	2,46 %
Paraguay	809	2,01 %
Otras	11.088	27,60 %

Ilustración 56: Instalaciones de usuarios por país.

Asimismo también han sido reportados los modelos de los dispositivos en los que ha sido instalada más veces la aplicación, estos son:



Instalaciones totales de usuarios de dispositivo

Samsung Galaxy S2 (G...	4.336	10,79 %
Samsung Galaxy S3 (m0)	3.392	8,44 %
Samsung Galaxy Y (GT-...	1.801	4,48 %
Samsung Galaxy Note (...	1.422	3,54 %
Samsung Galaxy Ace (...	1.108	2,76 %
Samsung Galaxy Mini (...	833	2,07 %
t03g	831	2,07 %
Samsung Galaxy S Plu...	759	1,89 %
Samsung Galaxy Ace (...	756	1,88 %
Otras	24.933	62,07 %

Ilustración 57: Instalaciones por dispositivo.

Para finalizar el presente capítulo se invita a los lectores a que descarguen y prueben la aplicación, el enlace a *Google Play* es el siguiente:

https://play.google.com/store/apps/details?id=com.jinchux.hideseek2

8 CONCLUSIÓN

Empecé a trabajar en este Proyecto de Fin de Grado hace ya casi un año, y al final me ha llevado mucho más tiempo del que en un principio pude imaginar. Todo comenzó cuando realicé un trabajo sobre esteganografía para la asignatura de "Seguridad en dispositivos móviles", en el que me involucré en una profunda investigación sobre el tema. Ese trabajo despertó sobre mí un gran interés y decidí proponerle al profesor que impartía esa asignatura la realización del actual proyecto.

Meses después se han recogido los frutos de todo el esfuerzo y sacrificio realizado. El grado de implicación en este trabajo ha sido máximo por mi parte, por lo que me siento especialmente satisfecho al comprobar que la idea que se forjó inicialmente se ha podido cumplimentar con creces. La publicación de la aplicación en *Google Play* ha culminado el presente PFG, demostrando que el software desarrollado no sólo ha cumplido los objetivos marcados sino también ha podido servir de utilidad a más de 40000 usuarios que han usado la aplicación por todo el mundo.

He aprendido mucho durante el desarrollo de este trabajo, desde la creación e implementación de aplicaciones para la plataforma Android hasta la adquisición de conocimientos avanzados de esteganografía y criptografía. Como marco de fondo siempre ha estado presente la seguridad como base fundamental para la construcción del sistema.

8.1 OBJETIVOS

El objetivo principal del proyecto fue desde el principio desarrollar una aplicación para Android que permitiera mantener comunicaciones seguras y discretas a través de la mensajería móvil tradicional y la instantánea, todo ello empleando la esteganografía como pilar fundamental. Dicho objetivo se ha cumplido, además se destacan los siguientes sub-objetivos alcanzados:

- La aplicación permite:
  - Ocultar mensajes en imágenes.
  - Enviar las imágenes a través de WhatsApp, Line, Spotbros, email y MMS.
  - Extraer mensajes ocultos de imágenes.
  - Actualizar la aplicación.
- La interfaz diseñada es bastante sencilla de usar, habiendo adaptado el modelo de asistente guiado para completar los pasos necesarios para ocultar y extraer mensajes.
- Se han añadido capas de seguridad adicionales para incrementar la protección de la información ocultada, esto incluyó algoritmos y funciones criptográficas.
- Se han desarrollado algoritmos que permiten dispersar de manera pseudo-aleatoria el mensaje oculto a lo largo de la imagen, haciendo más complicada su detección.
- Se han implementado mecanismos de recuperación de errores basados en códigos de redundancia, como los *Hamming*, que permiten reconstruir el mensaje original a pesar de haber sufrido daños durante su transmisión.
- Se ha conseguido ocultar mensajes de gran longitud en imágenes, independientemente del formato o de la resolución que posean.

- El grado de indetectabilidad alcanzado tras realizar pruebas de estegoanálisis ha sido muy elevado, hasta en un 85% de los casos en los que se han analizado imágenes generadas con la aplicación se han obtenido resultados satisfactorios.
- La seguridad completa del sistema se califica como muy notable, encontrándose que el único eslabón débil es el usuario.
- La velocidad y rendimiento de la aplicación han sido analizados en un dispositivo móvil de gama media con resultados muy aceptables.
- Se han obtenido resultados inimaginables tras la publicación de la aplicación en *Google Play* obteniendo un número de descargas que refleja la utilidad real del software desarrollado.
- La compatibilidad de la aplicación en la mayoría de dispositivos y versiones de Android ha sido corroborada gracias a los informes obtenidos a través de *Google Play* y los usuarios que han utilizado la aplicación.

Se puede concluir, por lo tanto, que se han alcanzado con éxito todos los objetivos que inicialmente fueron marcados, dejando abierta la posibilidad a futuras mejoras e implementaciones, cuestión que se comenta en la siguiente sección.

8.2 FUTUROS TRABAJOS

Tras el éxito cosechado publicando la aplicación se abren nuevas líneas de trabajo futuras para mejorar el funcionamiento de la aplicación e incorporar nuevos módulos funcionales.

Se ha pensado extender la compatibilidad del sistema a otros programas de mensajería, entre los ellos se ha barajado la posibilidad de enviar imágenes esteganografiadas a través de la aplicación "*Facebook Messenger*", lo que aumentaría el número de descargas actuales dado el elevado número de usuarios que utilizan ese sistema de mensajería. Asimismo sería interesante estudiar qué tratamiento reciben las imágenes cuando se suben a las redes sociales, esto incluye parámetros de compresión y escalado de imágenes, para poder hacer extensible la esteganografía a *Facebook* y *Tuenti* entre otros.

Se han recibido muchos correos electrónicos de usuarios demandando la necesidad de portar la aplicación a *Iphone,* cosa que no resultaría trivial dada la diferencia entre la plataforma Android e *iOS*. Otra posibilidad es la de exportar la aplicación JAVA para que pudiera usarse desde Windows, Linux o Mac, para lo cual habría que sustituir algunos métodos nativos de Android por otros equivalentes de JAVA y realizar una interfaz alternativa para el usuario.

Las aplicaciones esteganográficas del mercado que tienen más renombre ocultan la información en el dominio frecuencial de las imágenes, sin embargo se han hecho pruebas que revelan la incapacidad de esas imágenes generadas para conservar la información ocultada tras una compresión. Nuestro sistema proponía un enfoque distinto desde el dominio espacial, que precisamente es lo que permite la posibilidad de conservar los mensajes tras la compresión, siendo sacrificada levemente la calidad subjetiva de la imagen final. Por ello se propone también como posible mejora la optimización del sistema anterior, llegándose a dar la posibilidad de repartir un mensaje oculto en varias imágenes que posteriormente deban ser juntadas de nuevo para extraerlo. En todo caso las posibilidades son infinitas, sólo el ingenio pone el límite.

9 BIBLIOGRAFÍA

[1] *Criptografía*: http://laredargen.www6.50megs.com/criptografia.htm

[2] Johnson, Neil F.: *Steganography*: http://www.jjtc.com/stegdoc/index2.html

[3] La Biblia del Hacker, edición 2012.

[4] Silvia Torres Maya: *Tesis doctoral*.

[5] Arturo Ribagorda, Juan M. Estévez-Tapiador, Julio César Hernández: *Esteganografía, estegoanálisis e Internet* (2007).

[6] *CyberSpeak*: http://usatoday30.usatoday.com/tech/columnist/2001/12/19/maney.htm (2001).

[7] Death Master: *Introduction to Steganography*. http://www.death-master.tk/

[8] Inteco: *Esteganografía, el arte de ocultar información*.

[9] Adel Almohammad: *Steganography-Based Secret and Reliable Communications: Improving Steganographic Capacity and Imperceptibility* (tesis doctoral).

[10] Katzenbeisser and Petitcolas: *Information Hiding Techniques for Steganography and Digital Watermarking* (2000).

[11] Leezaben Ashokkumar Patel: *Steganography using cylinder insertion algorithm and mobile based stealth steganography* (2010).

[12] O. Marquest: *Image data representations: Binary images, gray-level images* (2010). http://encyclopedia.jrank.org/articles/pages/6761/Image-Data-Representations.html

[13] M.M. Sathik and N.R.S. Parveen: *Feature extraction on colored x-ray images by bit plane slicing techniques* (2010).

[14] *Compresión de imagen*: http://www.mvnet.fi/index.php?osio=Tutkielmat&luokka=Yliopisto&sivu=Image_compression

[15] V. Lokeswara Reddy: *Implementation of LSB Steganography and its Evaluation for Various File Formats* (2011).

[16] Philip Bateman: *Image Steganography and Steganalysis* (2008).

[17] *Los Mejores Sistemas Operativos Para Dispositivos Móviles: Lo Último en Tecnología:* http://zoominformatico.com/los-mejores-sistemas-operativos-para-dispositivos-moviles-lo-ultimo-en-tecnologia-de-punta (2012).

[18] Google Play: https://play.google.com/store/search?q=steganography&c=apps&sort=1

[19] BOE-A-1999-23750: *Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal* (13 de Diciembre de 1999).

[20] *Ley Orgánica de Protección de Datos*: http://www.gestedsl.com/lopd_definicion.php

[21] BOE-A-2003-20253: *Ley General de Telecomunicaciones* (3 de Noviembre de 2003).

[22] BOE-A-1996-8930: *Ley de la Propiedad Intelectual* (22 abril 1996).

[23] Métrica versión 3: *Análisis del Sistema de Información*.

[24] *Código fuente de la aplicación de mensajería multimedia (MMS) para Android*: https://android.googlesource.com/platform/packages/apps/Mms/

[25] Fernando Acero: *el algoritmo AES*. http://fernando-acero.livejournal.com/78069.html

[26] *Códigos Hamming*: http://www.kean.edu/~asetoode/home/crc2/hamming.htm

[27] *Seguridad de AES contra fuerza bruta:* http://www.eetimes.com/design/embedded-internet-design/4372428/How-secure-is-AES-against-brute-force-attacks-

[28] *Ataque de fuerza bruta*: http://erratasec.blogspot.com.es/2011/06/password-cracking-mining-and-gpus.html

[29] Android Excellence: http://www.androidexcellence.com/2013/01/10/mensajes-secretos-en-whatsapp/

10 ANEXOS

10.1 ANEXO A: GLOSARIO

Se recogen en esta sección los acrónimos y términos empleados en el documento, ordenados alfabéticamente y acompañados de su definición.

Adb (Android Debug Bridge): Es una línea de comandos muy versátil que nos permite comunicarnos con un dispositivo Android conectado al equipo.

Bash (Bourne Again Shell): Es el intérprete de comandos más famoso de las distribuciones Linux.

BER (Bit Error Rate): Es la medida que empleada para conocer el número de errores que se producen durante la transmisión de una cierta cantidad de bits.

Bitmap: Es un mapa de bits que representa una imagen.

Boolean: Tipo básico de JAVA que representa dos posibles valores: *true* (verdad) y *false* (falso).

BPP (Bits por píxel): Medida utilizada para indicar el número de bits que son empleados por cada píxel, se corresponde con la profundidad de la paleta de colores de la imagen.

Bruteforce: También conocido como ataque de fuerza bruta. Emplea la búsqueda exhaustiva como estrategia para romper un sistema criptográfico.

Crackear: Acción de vulnerar o romper un mecanismo de seguridad que protege un sistema.

Dropbox: Plataforma de almacenamiento de archivos “en la nube”.

GHz: Es la medida de frecuencia empleada para medir la velocidad de un procesador.

Gnome: Es un entorno de escritorio para sistemas operativos Unix y derivados.

Hash: Un hash es una cadena de longitud fija resultado de aplicar una función resumen sobre un conjunto de elementos de entrada.

IDE (Entorno de Desarrollo Integrado): Es un entorno de programación compuesto por un conjunto de herramientas dedicadas a la elaboración de programas informáticos.

Integer: Tipo básico de JAVA que representa números enteros en un rango de 2^{32} valores.

Intent: Mecanismo de comunicación básico empleado en la programación Android para realizar llamadas entre los distintos componentes que forman las aplicaciones y el sistema operativo.

iOS: Es el sistema operativo empleado por Apple para desarrollar sus productos móviles.

Log: Es un registro de eventos producidos durante un rango de tiempo que permite controlar la ejecución de un programa o sistema.

Loosy: Se denomina *loosy* a la compresión realizada sobre un medio digital que produce la pérdida irreversible de información en el mismo.

Padding: Es un relleno que se aplica sobre un elemento o estructura informática con el fin de aumentar su tamaño.

Píxel: En el contexto de los medios digitales representa la menor unidad homogénea de color.

Plugin: Se trata de un complemento específico para un programa informático que permite aumentar su funcionalidad.

Rijndael: También conocido como AES, es uno de los algoritmos más populares utilizados en criptografía simétrica.

Salt: Es un dato aleatorio empleado como entrada adicional en funciones resumen o de derivación de contraseñas.

String: Tipo básico de JAVA que permite almacenar y representar cadenas.

Token: Es una cadena de caracteres que posee un significado específico en un lenguaje de programación.

10.2 ANEXO B: MANUAL DE USUARIO

10.2.1 INSTALACIÓN DE LA APLICACIÓN

Para instalar la aplicación el usuario debe:

- Abrir la aplicación *Play Store* (*Google Play)* desde su dispositivo móvil Android.
- En el campo de búsqueda escribir "mensajes secretos" o "esteganografía".
- Seleccionar la aplicación llamada **Mensajes secretos para Android** (figura 58).
- También se puede acceder a la página de descarga a través del siguiente enlace: **https://play.google.com/store/apps/details?id=com.jinchux.hideseek2**
- Pulsar el botón instalar en la pantalla de descripción de la aplicación (figura 59).

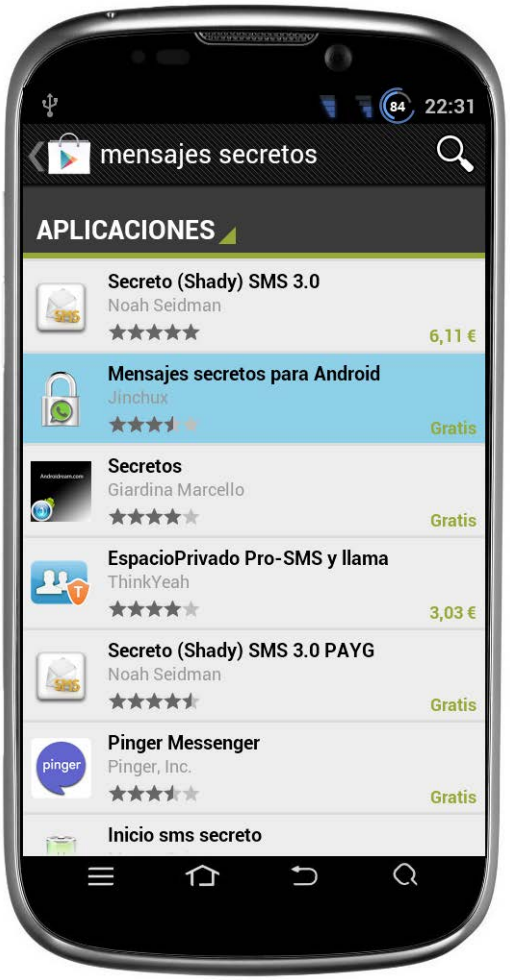


Ilustración 58: Instalación de la aplicación 1.

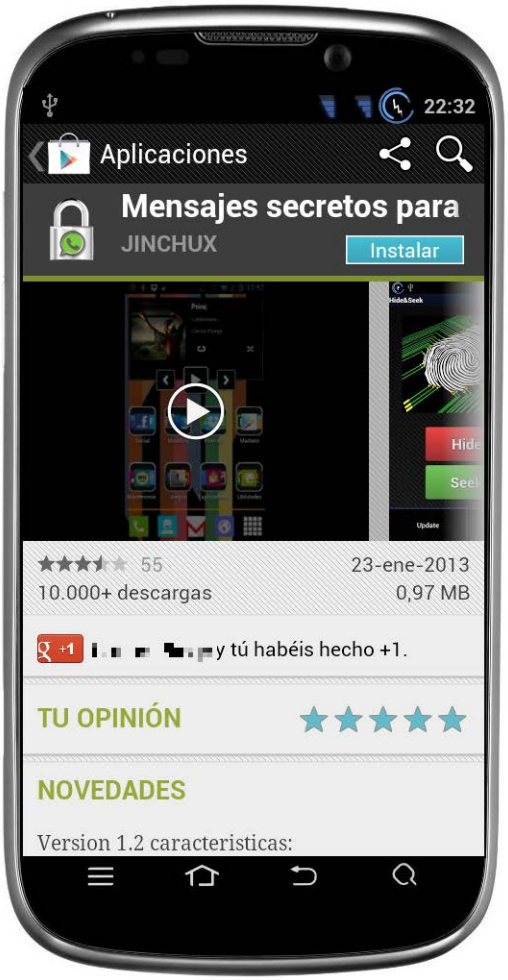


Ilustración 59: Instalación de la aplicación 2.

Una vez instalada estará disponible desde el menú de aplicaciones del dispositivo móvil. En la imagen 60 se muestra el icono de la aplicación. Es importante destacar que el nombre interno de la aplicación es "***Hide&Seek***", un juego de palabras que describe los dos modos de funcionamiento principales que posee la aplicación: "Ocultar" y "Encontrar".

Ilustración 60: Icono de aplicación *Hide&Seek*.

10.2.2 INSTRUCCIONES DE USO

OCULTAR MENSAJE

Tras haber completado el sencillo proceso de instalación y abrir la aplicación, se muestra una pantalla desde la que podemos: Ocultar un mensaje o extraer un mensaje de una imagen. Además pulsando el botón de menú nos aparecerán 2 botones por pantalla, desde los que se busca actualizaciones o se muestra la información de la aplicación. En la imagen 61 se muestra dicha pantalla principal.

Ilustración 61: Pantalla principal de la aplicación.

Ilustración 62: Pantalla de ocultación.

En la ilustración 62 se muestra la pantalla del modo de ocultación. En ella se deben rellenar todos los datos requeridos, que son:

1. Seleccionar una imagen de la galería.
2. Escribir el mensaje secreto que se desea ocultar.
3. Introducir una contraseña de cifrado, que será la misma que se emplee para descifrar.
4. Escoger si la imagen resultante se prefiere en color o en blanco y negro.
5. Se elige la aplicación del listado a la que queramos enviar la imagen generada.

Tras completar todos los campos, hay que presionar sobre el botón “Ocultar”, entonces comenzará la ocultación.

EXTRAER MENSAJE

Para acceder al modo de extracción de un mensaje ocultado en una imagen, desde la pantalla principal hay que pulsar el botón “Encontrar”. Tras ello tendremos que rellenar los campos que la aplicación propone, éstos son:

1. Seleccionar una imagen que contiene un mensaje oculto.
2. Introducir la contraseña con la que se ocultó el mensaje.

En la imagen 63 se muestra la pantalla con los datos que se tienen que completar. Una vez se han introducido éstos se puede pulsar el botón “Descifrar” para ver el mensaje oculto.

Ilustración 63: Pantalla de extracción del mensaje.

Ilustración 64: Pantalla con mensaje extraído.

En la figura 64 se aprecia el resultado tras presionar el botón “Descifrar”, se muestra el mensaje que originalmente fue ocultado.

ENVÍO A WHATSAPP

Cuando se desea enviar una imagen oculta a WhatsApp hay un modo más rápido de acceder a la aplicación. Hay que seguir los siguientes pasos:

1. En la pantalla de chat pulsar sobre el icono de adjuntar fichero (clip).
2. Seleccionar opción “Galería” como se muestra en la figura 65.
3. Seleccionar modo “Ocultar secreto” tal y como se aprecia en la imagen 66.

Ilustración 65: Menú adjuntar de WhatsApp.

Ilustración 66: Seleccionar imagen desde programa.

La ventaja que tiene abrir la aplicación desde el chat de WhatsApp es que accedemos directamente al modo de ocultación de “*Hide&Seek*”. Además no es necesario introducir el modo de envío, pues si hemos accedido desde WhatsApp la imagen se mandará ahí. Finalmente la imagen se enviará al contacto desde el que hemos llamado a la aplicación.

RECEPCIÓN DE IMAGEN

Cuando se recibe una imagen esteganografiada en WhatsApp, Line o Spotbros, es sencillo abrir directamente la imagen con "*Hide&Seek*" para extraer su mensaje. Dependiendo de en cuál de las 3 aplicaciones nos encontremos se aplicará un procedimiento distinto.

- Para abrir la aplicación desde WhatsApp basta con pulsar encima de la imagen recibida hasta que aparezca el menú mostrado en la figura 67.
- Para llegar al mismo menú desde Line es necesario pulsar en la imagen una vez, cuando se haya abierto en pantalla completa basta con pulsar el botón de "Menú" de Android y seleccionar la opción "Usar otra aplicación" como se muestra en la imagen 68.
- Para realizar la misma acción desde Spotbros hay que descargar en primer lugar la imagen desde el programa. Después basta con pulsar el botón "Ampliar" debajo de la imagen a la derecha. En la ilustración 69 muestra este sencillo proceso.

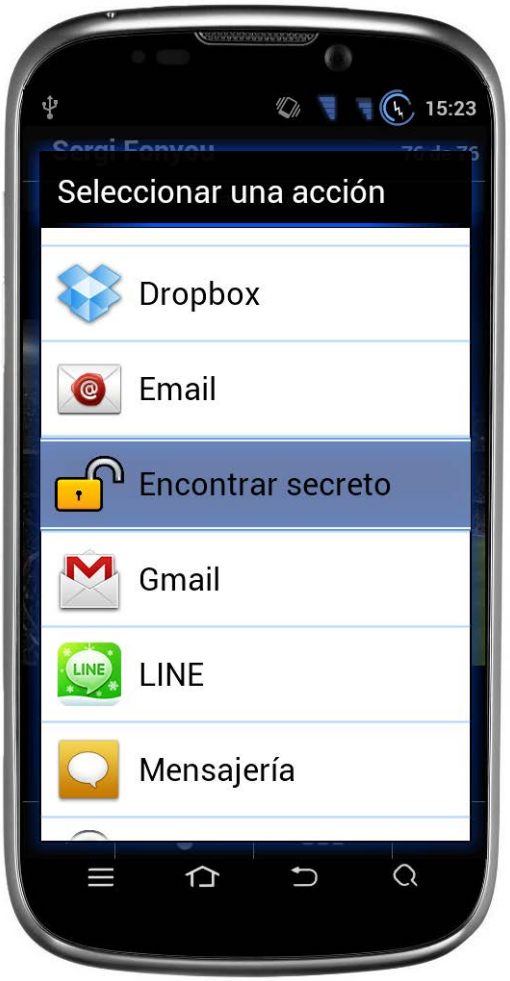


Ilustración 67: Menú de selección de programa.

Ilustración 68: Acceder al menú de programa desde Line.

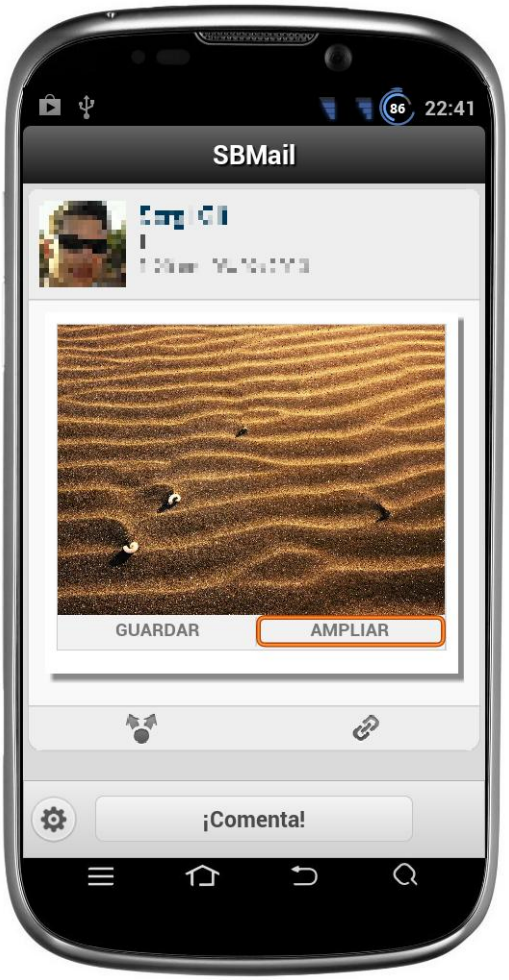


Ilustración 69: Botón "Ampliar" de Spotbros.

Para el caso de los correos electrónicos y los MMS el proceso es más largo. De hecho, no se puede abrir directamente la imagen hasta después de haberla descargado. Así que será necesario abrir el programa desde el icono de la aplicación y buscar en la galería la imagen descargada, que por lo general se encontrará en la carpeta “Descargas”.

ACTUALIZACIÓN DE LA APLICACIÓN

Cuando haya una actualización nueva disponible, siempre que dispongamos de una conexión a Internet, la aplicación nos informará de ello automáticamente tras entrar en el programa.

En primer lugar nos aparecerá un mensaje por pantalla en el que se describen los cambios nuevos introducidos en la nueva versión de la aplicación respecto a la anterior. En la ilustración 70 se observa el mensaje de novedades para la última versión de "*Hide&Seek*".

Ilustración 70: Mensaje de novedades de actualización.

Ilustración 71: Diálogo de descarga de nueva versión.

En la figura 71 se muestra el mensaje posterior a la pantalla de novedades, en la cual se propone al usuario la descarga de la última versión de la aplicación. Dicha descarga se efectuará siempre que se pueda desde la interfaz de *Google Play*.

VÍDEO-TUTORIAL EXPLICATIVO

Se ha desarrollado un vídeo explicativo que indica los pasos que hay que seguir para:

- Ocultar y enviar desde WhatsApp una imagen esteganografiada con "*Hide&Seek*".
- Extraer el mensaje oculto de la imagen recibida.

El tutorial muestra la perspectiva del emisor y la del receptor del mensaje, de modo que se aprecia mejor el funcionamiento de la aplicación. Las instrucciones están en inglés pero la aplicación se muestra en castellano en el vídeo.

El vídeo ha sido colgado en *Youtube*, el enlace para ver el tutorial es el siguiente: http://youtu.be/nQlFIESKK_0

A pesar de que el tutorial sólo muestra un ejemplo de envío y recepción en WhatsApp, el usuario se hace una idea básica de la funcionalidad del producto.

10.3 ANEXO C: PRESUPUESTO

UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1. Autor:

Sergio Pérez Olivares

2. Departamento:

Departamento de Informática

3. Descripción del Proyecto:

Título **Desarrollo de una aplicación esteganográfica para Android**

Duración (meses) **11**

Tasa de costes Indirectos: **20%**

4. Presupuesto total del Proyecto (valores en Euros):

14.715,00 Euros

5. Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar sólo a título informativo)	Categoría	Dedicación en horas de trabajo	Coste por hora	Coste (Euro)
Pérez Olivares, Sergio		Jefe de Proyecto	60	30,00	1.800,00
Pérez Olivares, Sergio		Analista	150	22,00	3.300,00
Pérez Olivares, Sergio		Diseñador	70	16,00	1.120,00
Pérez Olivares, Sergio		Programador	300	12,00	3.600,00
Pérez Olivares, Sergio		Documentalista	90	18,00	1.620,00
				Total	11.440,00

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Intel Centrino 2 2.26 GHz	600,00	100	10	60	100,00
Samsung Galaxy S	150,00	100	10	60	25,00
				Total	125,00

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

OTROS COSTES DIRECTOS DEL PROYECTO [e)]

Descripción	Empresa	Costes imputable
Internet	Ono	60,00
Luz	Iberdrola	80,00
Cuenta desarrollador Android	Google	18,50
Microsoft Office 2013	Microsoft	539,00
	Total	697,50

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas.

6. Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	22.080
Amortización	125
Subcontratación de tareas	0
Costes de funcionamiento	698
Costes Indirectos	2.453
Total	**14.715**

10.4 ANEXO D: PLANIFICACIÓN

A continuación se representa la **planificación inicial** estimada marcando sólo los procesos principales:

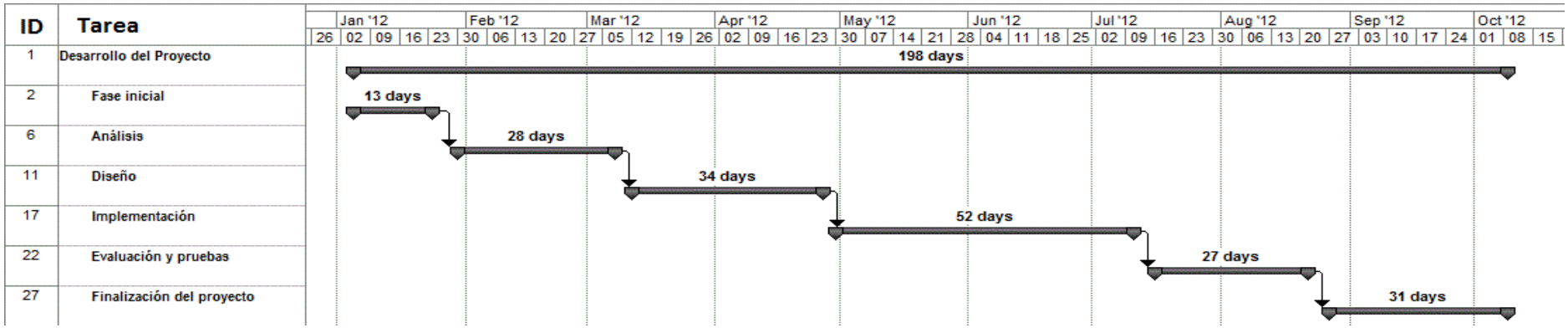


Ilustración 72: Planificación inicial simplificada.

La **planificación final** se muestra a continuación:

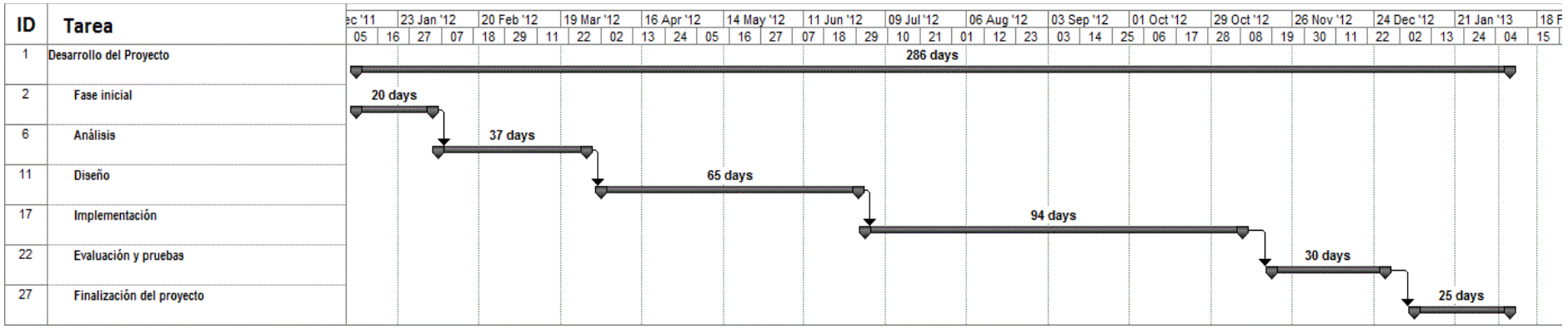


Ilustración 73: Planificación final simplificada.

A continuación se desglosan las tareas que componen cada uno de los procesos del proyecto para la planificación inicial:

ID	Tarea
1	**Desarrollo del Proyecto**
2	**Fase inicial**
3	Proposición del proyecto
4	Definición de objetivos
5	Planificación del proyecto
6	**Análisis**
7	Estado del arte
8	Definición del sistema
9	Análisis de requisitos
10	Modelo de casos de uso
11	**Diseño**
12	Diseño de arquitectura
13	Diseño de subsistema de ocultación
14	Diseño de subsistema de extracción
15	Diseño de subsistema de actualización
16	Diseño de interfaces de usuario
17	**Implementación**
18	Ingeniería inversa de aplicaciones
19	Implementación del sistema de cifrado
20	Implementación del sistema de ocultación
21	Implementación de recuperación de errores
22	**Evaluación y pruebas**
23	Pruebas de esteganografía
24	Pruebas de rendimiento
25	Pruebas de seguridad
26	Publicación de la aplicación
27	**Finalización del proyecto**
28	Conclusiones
29	Presupuesto
30	Documentación

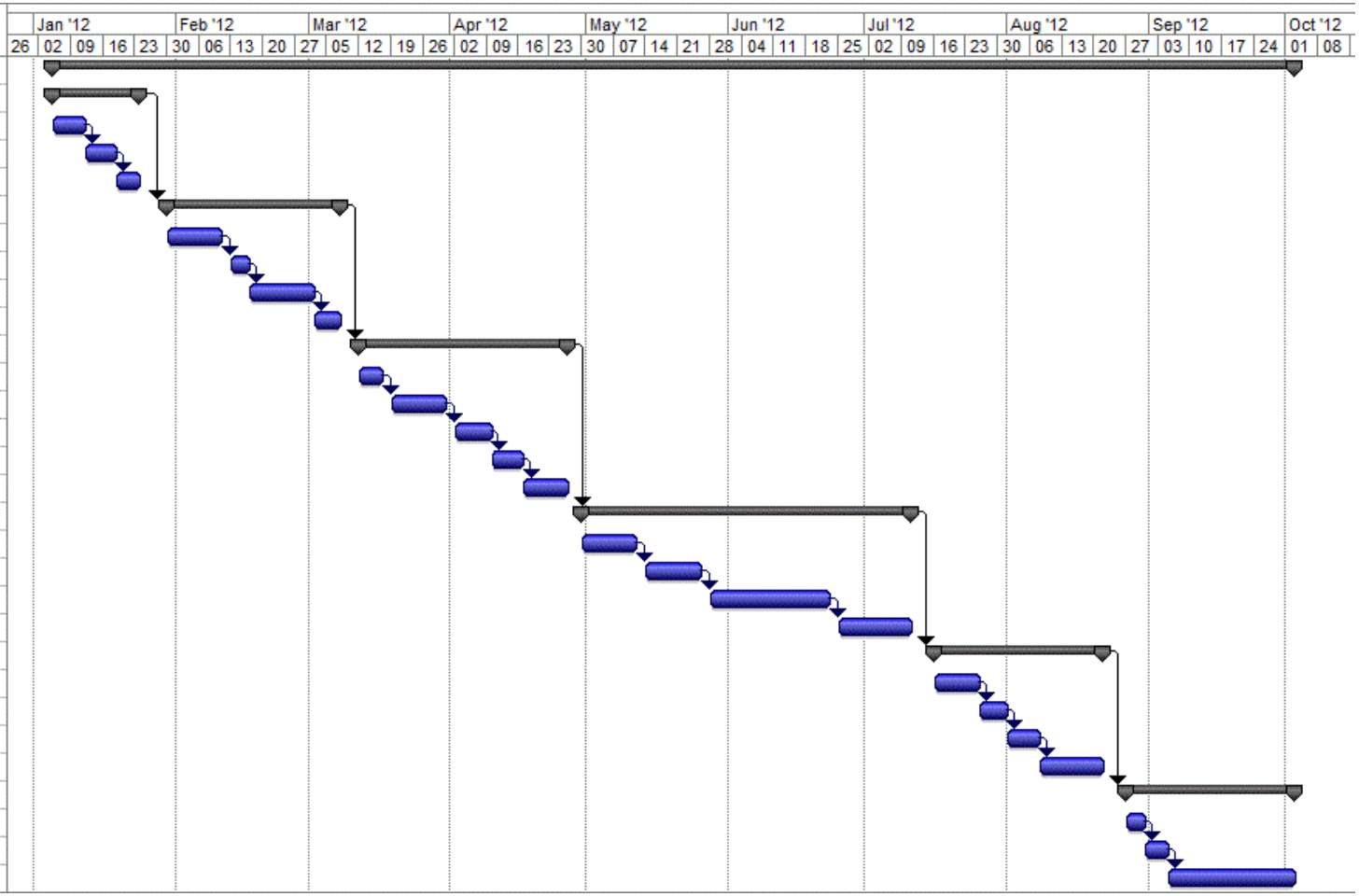


Ilustración 74: Planificación inicial extendida.

En el siguiente diagrama de Gantt se desglosan los procesos que componen la planificación final del proyecto:

ID	Tarea
1	**Desarrollo del Proyecto**
2	**Fase inicial**
3	Proposición del proyecto
4	Definición de objetivos
5	Planificación del proyecto
6	**Análisis**
7	Estado del arte
8	Definición del sistema
9	Análisis de requisitos
10	Modelo de casos de uso
11	**Diseño**
12	Diseño de arquitectura
13	Diseño de subsistema de ocultación
14	Diseño de subsistema de extracción
15	Diseño de subsistema de actualización
16	Diseño de interfaces de usuario
17	**Implementación**
18	Ingeniería inversa de aplicaciones
19	Implementación del sistema de cifrado
20	Implementación del sistema de ocultación
21	Implementación de recuperación de errores
22	**Evaluación y pruebas**
23	Pruebas de esteganografía
24	Pruebas de rendimiento
25	Pruebas de seguridad
26	Publicación de la aplicación
27	**Finalización del proyecto**
28	Conclusiones
29	Presupuesto
30	Documentación

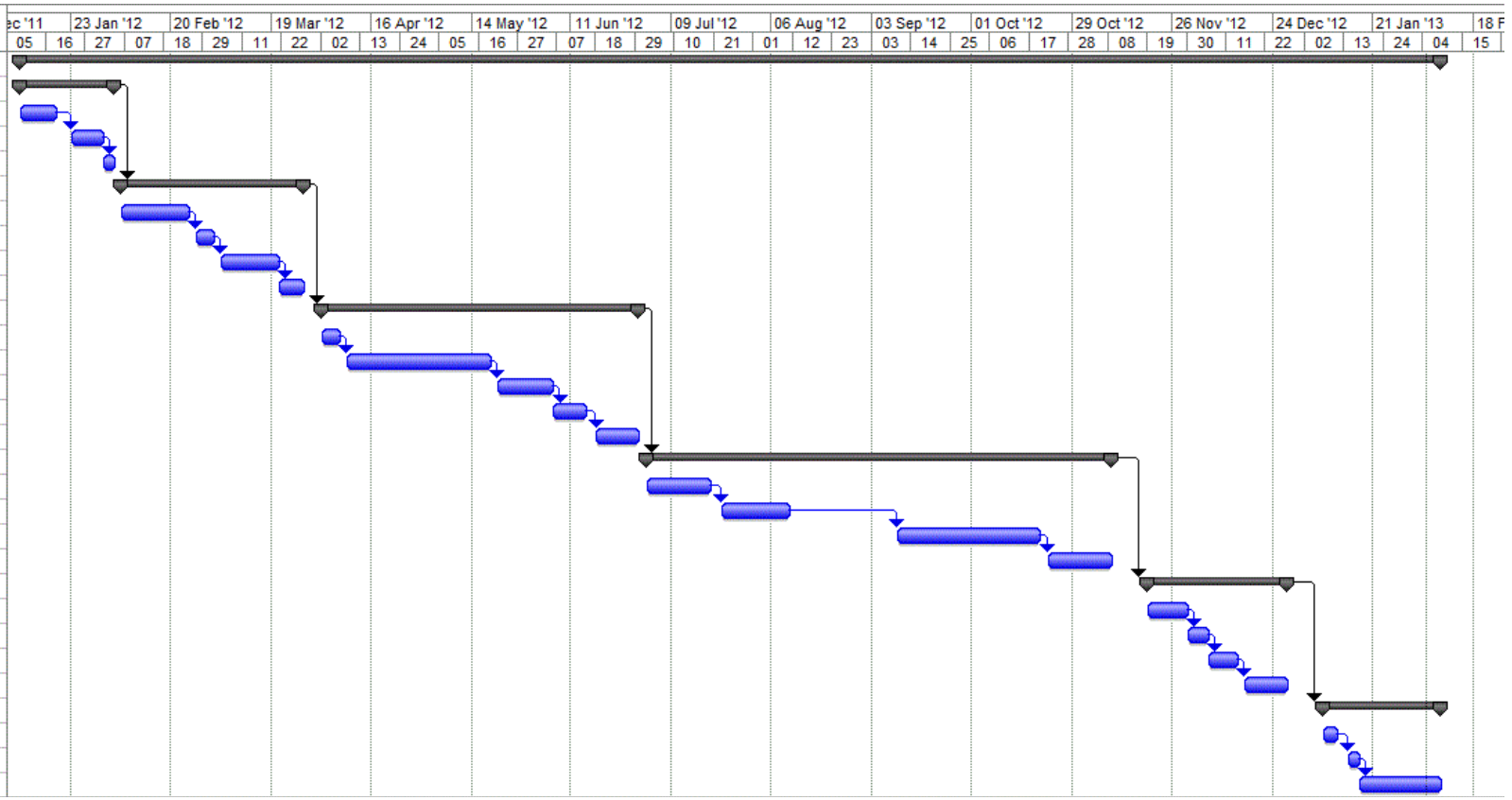


Ilustración 75: Planificación final extendida.

ANÁLISIS DE DESVIACIONES

Comparando los diagramas de planificación inicial y final, se puede observar que se ha producido una importante desviación entre ambos. En principio se estimó que 198 días serían suficientes para completar el desarrollo del proyecto actual. Sin embargo, al final fueron necesarios 286 días, es decir, 178 días más de lo esperado.

En los diagramas de planificación detallados (figuras 74 y 75) se aprecia que la principal desviación se produce en las actividades de "Diseño" e "Implementación". Los motivos más significativos por los que se ha producido dicha desviación se explican brevemente a continuación:

- Se ha invertido una gran cantidad de tiempo en la investigación y búsqueda de documentación relacionada con la esteganografía y proyectos similares.
- La inexperiencia en el campo de la esteganografía y del desarrollo de aplicaciones para dispositivos móviles Android.
- Ha sido necesario mucho tiempo para realizar pruebas durante el diseño del sistema de ocultación.

Las razones anteriores no fueron consideradas inicialmente, por ello la planificación real difiere tanto de la estimada en principio.